2006-09

A prototype implementation of a time interval file protection system in Linux

Chiang, Ken H.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A PROTOTYPE IMPLEMENTATION OF A TIME INTERVAL FILE PROTECTION SYSTEM IN LINUX

by

Ken H. Chiang

September 2006

Thesis Advisor:	Cynthia E. Irvine
Co-Advisor:	Thuy D. Nguyen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: A Prototype Implementation of a Time Interval File Protection System in Linux			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ken H. Chiang			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Control of access to information based on temporal attributes has many potential applications. Examples include student user accounts set to expire upon graduation; files marked as time-sensitive so that their contents can be protected appropriately and the period of access to them controlled; and cryptographic keys configured to automatically expire and be unusable beyond a specific time. This thesis implements a prototype of the Time Interval Access Control (TIAC) model in the context of a protected file system for the popular open-source Linux operating system. The Linux Security Module framework is used for the implementation, which includes temporal attributes associated both with the files and the users.

The implementation includes modifications to the file system as well as low-level information access constructs. As part of the design process, testing and performance analysis were conducted.

Since the temporal access control mechanism is built into the kernel rather than the application, bypassing the mechanism becomes more difficult. Kernel level implementation also affords the same policy enforcement functionality to different applications, thus reducing human errors in their development. This thesis is relevant to the research on dynamic security services for information protection envisioned by the DoD Global Information Grid (GIG).

14. SUBJECT TERMS Temporal Access Control, Linux Kernel, Linux Security Module, Dynamic Security Services, Global Information Grid			**15. NUMBER OF PAGES** 246
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A PROTOTYPE IMPLEMENTATION OF A TIME INTERVAL FILE PROTECTION SYSTEM IN LINUX

Ken H. Chiang
Civilian, Federal Cyber Corps
B.S., Georgia Institute of Technology, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2006

Author: Ken H. Chiang

Approved by: Cynthia E. Irvine, PhD
Thesis Advisor

Thuy D. Nguyen
Co-Advisor

Peter J. Denning, PhD
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Control of access to information based on temporal attributes has many potential applications. Examples include student user accounts set to expire upon graduation; files marked as time-sensitive so that their contents can be protected appropriately and the period of access to them controlled; and cryptographic keys configured to automatically expire and be unusable beyond a specific time. This thesis implements a prototype of the Time Interval Access Control (TIAC) model in the context of a protected file system for the popular open-source Linux operating system. The Linux Security Module framework is used for the implementation, which includes temporal attributes associated both with the files and the users.

The implementation includes modifications to the file system as well as low-level information access constructs. As part of the design process, testing and performance analysis were conducted.

Since the temporal access control mechanism is built into the kernel rather than the application, bypassing the mechanism becomes more difficult. Kernel level implementation also affords the same policy enforcement functionality to different applications, thus reducing human errors in their development. This thesis is relevant to the research on dynamic security services for information protection envisioned by the DoD Global Information Grid (GIG).

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Success is rarely accomplished by individuals alone. My successful completion of this thesis is no exception. This work is not complete without the acknowledgement of the following:

Dr. Cynthia Irvine and Professor Thuy Nguyen, thank you for providing me with a guiding light throughout this journey and keeping me on track as my advisors.

Mr. Phil Hopfner, thank you for providing the computing resources and support necessary for completing my thesis.

Mr. Jean Khosalim, thank you for the many hours spent on test procedure verification and for providing feedback for improvements.

Mr. John Clark, thank you for being my sounding board in discussion of issues and ideas throughout this research.

Ms. Tanya Raven, thank you for being a friend and a believer in me. Without your support and encouragement, I may not have applied to the Federal Cyber Corps scholarship program at the Naval Postgraduate School and would have missed this great opportunity.

Finally, Sherry Chiang, my beloved wife, thank you for being there for me throughout my pursuit for higher education. Your understanding and support makes the pursuit of my goals possible and your unwaivering love makes reaching these goals all the more worthwhile.

This material is based upon work supported by the National Science Foundation under grant No. DUE-0114018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Controlling access to information based upon time constraints has potential applications in government, military, financial, and educational realms. Temporal access control can permit access to information based upon a start time and revoke access based upon a stop time. For example, in the government and military, access to cryptographic keys used to encrypt information can expire at a certain time to further protect the confidentiality of the information. In the financial realm, organizations award individuals incentives in the form of stock options to motivate its employees. Typically, these incentives are time sensitive, i.e. they cannot be redeemed until a certain time in the future. Finally, in education, there is a constant flux of incoming and exiting students. Providing availability to access computer resources and controlling such access based upon the time during which the students are enrolled is a task simplified with temporal access control.

The Global Information Grid envisions networks of computing systems that enable global sharing and proper control of information through dynamic security services. Time-based access control systems can support dynamic security services by changing access permissions based upon time. The capability of such a system to grant or revoke access at a future time as well limiting access to information to a specific time interval can provide a new control vector for information sharing not available in traditional access control systems.

In a computer system, there is more than one component into which a time-based access control mechanism can be built. Two such components are the application and the operating system. If the mechanism resides in the operating system, it will be much harder for a malicious user to bypass the mechanism. Since all applications depend on fundamental system services provided by the operating system, i.e. device read, write, etc, the operating system can be a focal point of control for many applications that need access to system resources. This centralized access control minimizes the complexity of developing a complete set of applications attempting to enforce a time-based policy and

thus results in better security. For example, consider the scenario where the access control mechanism is built into only one application, if the user can copy the information into another application where no such mechanism is in place, he will have successfully bypassed the access control mechanism. Also, to a more sophisticated attacker, bypassing the access control mechanism at the application level could be as easy as creating his own application to access the information. This thesis explores a prototype implementation of temporal access control in an operating system.

B. PURPOSE

Afinidad et al. described a Time Interval Access Control (TIAC) model in which time-based access control is formally modeled using interval algebra [1, 3]. Here, an implementation of this model is prototyped in the popular Linux operating system. This work helps to answer the following questions:

- What specific changes are necessary to the Linux kernel to implement TIAC model for file access?
- What practical design implications are there for building such a system?

Additionally, this prototype will serve as a baseline for performance evaluations of future implementations of time-based systems. To establish this baseline, the performance overhead of this implementation will be compared with the performance of an unmodified Linux operating system. Finally, this prototype may serve as a basis for exploring user acceptability of TIAC.

C. ORGANIZATION OF THESIS

This thesis is organized as follows:

- This chapter (Chapter I) provided an introduction by describing the motivation and purpose of the thesis. The TIAC model was briefly introduced and serves as a basis for this study.
- Chapter II provides a more detailed description of the TIAC model. It also introduces the Temporal Interval Memory Protection System (TIMPS) which was a study of an application of TIAC on memory at the hardware level. The Linux operating system's file management system is described next to provide background for the envisioned implementation of TIAC. Finally, Linux command line utilities and the Command Line Interface (CLI) used to interact with and test the system are described. The CLI described provides a basis for the envisioned implementation of the tool for interacting with the time-based access control system.

- Chapter III gives a high level description of the requirements and design for the Time Interval File Protection System (TIFPS) and associated CLI tool used for interacting with the time-based access control system. This chapter also discusses implementation choices made during the research. Next, a description of the development environment is given. Finally, selected implementation details for the system are provided.
- Chapter IV describes the high level test plan and the analysis of the test results. Testing plan is divided into three categories: access control, performance, and concurrency testing.
- Chapter V concludes with a thesis summary and suggestions for short term and long term future work.
- The appendices follow with a listing of the TIFPS-related source code in Appendix A. The installation and usage guide for TIFPS are located in Appendix B and C, respectively. Appendix D captures the test procedures, scripts, and results based on the testing plan in Chapter IV. Finally, Appendix E provides configuration files used in the development environment.

D. SUMMARY

In this introductory chapter, we motivated this research by describing potential applications of a temporal access control system and justified the benefit of kernel-level access control compared with application-level access control. The organization of the thesis was then presented. We continue with the background of this research in Chapter II.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

This chapter provides background information that motivated and influenced the work performed in this research. It starts with a description of the Time Interval Access Control (TIAC) model which is a formal model that describes the authorization of access to objects based upon the time attributes of the subject and the object [1,3]. From the TIAC model, a time-based, hardware level, memory protection scheme call Temporal Interval Memory Protection System (TIMPS) was devised [2]. A brief discussion on how TIMPS works and our consideration of its use for this research follows. In this research, we will provide a design and implementation of the TIAC model applied to regular files and directories in a Linux operating system (kernel). Thus, we will give a description of the Linux file management system. Finally, Linux command line utilities that will be used or built to demonstrate time-based file and directory access control will be discussed.

A. TIME INTERVAL ACCESS CONTROL (TIAC) MODEL

To correctly implement a time based access control system, unambiguous semantics needs to be first developed to describe the desired security policies. The TIAC model is a formal mathematical model developed using interval algebra. This model associates time attributes with subject and object entities and describes access authorizations using the notion of access graphs. Using formal semantics to describe access policy gives us the ability to precisely decide, at any given time, when a subject with a given set of time attributes, has permissions to access an object, which also has time attributes. Since the model has only three time intervals, i.e. those associated with subject and object, and the time interval during which access is requested, access policies using this model can be checked for consistency using existing algorithms [1]. The details of the formal model are described in a recent paper by Afinidad et al. [1]. It is important to note that this model differs from previous models in that it supports policies based upon temporal attributes of subject and object rather than object alone.

B. TIME INTERVAL MEMORY PROTECTION SYSTEM (TIMPS)

Based on the TIAC model, Afinidad et al. also presented the Time Interval Memory Protection Systems (TIMPS) where all access to memory is mediated according

to time-based access control policy [2]. To understand how TIMPS works, we must first understand how memory management works in modern operating systems. To support multi-tasking, most modern operating systems (including Linux) use a memory management technique known as paging. In paging, physical memory is divided into chunks of equal size called page frames. Each running process in the operating system has its own virtual memory address space which consists of virtual memory chunks appropriately called pages. These virtual pages are mapped to the physical page frames by a memory management unit (MMU). The MMU keeps the address space of each process separate by mapping the virtual pages to different physical page frames. Therefore, when a process needs to access memory, a translation of the virtual page to the physical page frame must occur. The access control mechanism in the TIMPS protection schemes lies in the translation of virtual memory addresses to the corresponding physical memory addresses in the paging mechanism.

The work done on TIMPS previously was largely performance motivated and used hardware simulation to provide the necessary hardware support. Afinidad et al. designed, compared, and contrasted different schemes using a combination of hardware and software to implement time-based access control to memory. To help analyze the results, performance of the access control mechanism was divided into an initial authorization phase and an ongoing access phase. The initial authorization phase describes the access mediation of a new request by a subject process to a memory object. In this phase, temporal logic used to calculate the expiration time resides in either hardware or software. If access is allowed as a result of this calculation, the expiration time is set in appropriate hardware fields so that subsequent checks for ongoing access can occur by checking only the expiration time. The ongoing access phase describes the access mediation that occurs after a subject has been granted initial access to an object. In this phase, temporal logic is implemented in hardware to check access of memory addresses by using the expiration time calculated in the initial authorization phase. The results of the TIMPS study can be summarized as follows. For systems that tend to spend more time in the ongoing access phase rather than initial authorization (i.e. personal desktop computers, PDAs, laptops), computations related to calculating the initial expiration time of a memory chunk can reside in software since the performance benefit

of implementing the logic in hardware is negligible. For systems that spend a lot of time in the initial authorization phase (i.e. Servers), the study recommended that the temporal logic used to calculate the expiration time of allocated memory objects be implemented in hardware as an added module to the CPU rather than software.

In this research, we considered implementing TIMPS completely in software for the purpose of file access using an existing operating system, Fedora Core 5 running the Linux 2.6.15. The motivations for this software implementation are:

1. To provide a framework for future time-based access control systems in non-simulated environments.
2. To provide a baseline for future performance studies in true hardware environments.
3. To potentially provide a means to conduct user-acceptance studies of time-based access control systems.

However, upon a more detailed study of potential designs and implementation, we were hindered by a problem caused by the paging mechanism. To understand the problem, note that the granularity of memory access control is in pages that are typically 4K in size. Assume that we want to end access to the memory location where protected file content has been read. If this memory location is not page aligned and we deny access to the entire page, we will also be denying access to variables that may be needed by the process in order to run correctly. In this research, access control to files is the focus, therefore, file-level instead of memory-level granularity will be used for the design and implementation of the time-based access control system. It is important to clarify the meaning of "files" in this implementation. In Linux, almost everything is considered a "file"; directories, network sockets, devices, symbolic links, regular files in a mounted file system, etc. In this implementation, when a "file" or "regular file" is mentioned, it refers to a regular file in a mounted file system.

C. LINUX FILE MANAGEMENT

The Linux kernel implements a software layer that handles all system calls related to a standard Unix-based file system. This arrangement allows different file systems to coexist and interoperate on the Linux operating system and enables file operations on these different file systems independent of the file system type. The software abstraction is called the Virtual File System (VFS) and consists of four structures/objects. They are:

- The super block object, which describes information about the specific mounted file system and corresponds to the file system super-block or control block.
- The inode object, which contains information needed to manipulate a file, directories, and other file system objects. Access permissions, owner, group, and time information associated with the file are stored in this structure.
- The dentry object, which represents a directory entry, a single component of a path. For example, /bin/emacs consists of the following dentry objects: "/", "bin", and "emacs". It is important to note that directories are treated as files in Linux.
- The file object, which represents an open file associated with a process that opened it.

In addition to the objects described above for controlling access to files and directories, Linux has implemented Extended Attributes (EA) for most file systems starting with the 2.6 kernel. EAs are name/value pairs associated and stored permanently with files that allow additional control over how files are accessed. This feature enables a consistent means to extend file system capabilities and maintain file system independence. Security Enhanced Linux (SELinux), a flexible access control mechanism recently added to Linux by the NSA largely known for enforcing mandatory access control policies, uses EAs for labeling files. There are four predefined namespaces supported in the Linux 2.6.15 kernel. They are: security, system, trusted, and user. The following is a description of each of these namespaces.

Extended security attributes

The security attribute namespace is used by kernel security modules, such as Security Enhanced Linux. Read and write access permissions to security attributes depend on the policy implemented for each security attribute by the security module. When no security module is loaded, all processes have read access to extended security attributes, and write access is limited to processes that have the CAP_SYS_ADMIN capability.

Extended system attributes

Extended system attributes are used by the kernel to store system objects such as Access Control Lists and Capabilities. Read and write access permissions to system attributes depend on the policy implemented for each system attribute implemented by file systems in the kernel.

Trusted extended attributes

Trusted extended attributes are visible and accessible only to processes that have the CAP_SYS_ADMIN capability (the super user usually has this capability). Attributes in this class are used to implement mechanisms in user space (i.e., outside the kernel) which keep information in extended attributes to which ordinary processes should not have access, i.e. md5 checksums.

Extended user attributes

Extended user attributes may be assigned to files and directories for storing arbitrary additional information such as the mime type, character set or encoding of a file. The access permissions for user attributes are defined by the file permission bits.

In this research, extended security attributes will be used to label files and directories with temporal attributes.

D. LINUX COMMAND LINE UTILITIES

In this research, a temporal access control mechanism will be built into the Linux kernel for controlling file and directory access. To demonstrate and test the new kernel prototype functionality, standard Linux command line utilities will be used. It is also anticipated that new command line utilities will be built to interface with the time-based access control system. These command line utilities are discussed in this section.

Linux provides a set of standard system utilities for interacting with the system. These utilities are issued to the system via the Command Line Interface (CLI). For example, to display the contents of a text file, the command **cat** can be used. To display the contents of a directory, the command **ls** can be used. Each of these system commands

includes various options settable by flags. The usage instructions of the commands as well as the various flag options can be retrieved by using the **man** pages. For example, to see all the options for **cat**, type:

$ man cat

In this time-based access control system, if based on its temporal attributes, access to a file or directory has expired, the kernel should return an access-denied signal to the process and the system utility should subsequently return the appropriate error to the user on the console and quit. It is important to note that in this prototype, files and directories that do not have temporal attributes will have a default-permit access and be treated as if there is an infinite allowed time to access them. Therefore, the administrator account, ***root***, will need to explicitly set the time attributes of files or directories for which he wishes to control access.

To interface with the time-based system, we will build a simple command line utility which will run in the CLI described above. This utility will have different flag options so that users can view the time attributes and administrators can modify the time attributes of files or directories.

E. SUMMARY

In this chapter, the concept of time-based access control as described by the TIAC model was described. We also discussed previous work on a Time Interval Memory Protection System which is an implementation of the TIAC model for memory. These topics provided background for the research into a time-based access control system for files and directories. Since this research will attempt to build a prototype using the Linux kernel, an understanding of the Linux file management system is needed and therefore introduced. Finally, for demonstration and testing the time-based system, standard Linux command line utilities will be used. The same Command Line Interface in Linux used for running the standard system utilities will also be used for building custom tools for interacting with the time-based system.

III. DESIGN AND IMPLEMENTATION OF TIFPS

Using the Time Interval Access Control (TIAC) model as a reference, a Time Interval File Protection System (TIFPS), capable of providing time based access control to files, was designed and implemented. This chapter covers the requirements for such a system, a high level description of the design, choices made in the design, the details of the implementation, and a brief description of the development environment.

A. REQUIREMENTS

The following describes the requirements defined for the TIFPS kernel and the time attribute modification tool envisioned to be used for interacting with the TIFPS kernel.

1. TIFPS Kernel Requirements

- The kernel must protect and mediate all access to regular files and directories protected with time-of-allowed-access attributes. Time-based access control will be demonstrated on all file and directory reads, writes and executions.
- Modification of the time attributes associated with the file must be allowed only by the super user (administrator) account.
- The precision of time in revoking access to expired files should not be more than one second.
- The prototype will allow infinite access to files that have not been labeled with time attributes.
- On copy operations, the destination files must take on the most restrictive time attributes of the files read by the copying process. This will prevent information leakage.
- The administrator shall be able to set time-of-allowed access for subjects, i.e. user accounts, and objects, i.e. regular files and directories.

2. Requirements for the Time Attribute Modification Tool

In order to obtain, set, and modify the time attributes of the files in the system, a tool is required. The following is a list of requirements for such a tool.

- Though the system will enforce time-based policy based upon absolute time, i.e. on September 22, 2006 at 1700 hours revoke access to file.txt; time attributes shall be set by specifying them in either absolute time or relative time. Relative time shall be referenced from current time.

- The administrator interface shall be easy to use. For example, setting time attributes shall not require complicated calculations by the administrator.
- The tool shall be able to take multiple arguments to change or display the time attributes of multiple files and directories at once.
- Usage instructions shall be made readily available.
- If mistakes are made while using the tool, useful error messages shall be displayed to the user.
- The tool shall allow the user/admin to easily view the time attributes of files and directories.

B. HIGH LEVEL DESIGN

This section describes the high level design of the TIFPS kernel and the tool that will be used to interface with the system.

1. TIFPS Kernel High Level Description

Figure 3-1 shows the process flow diagram for a user accessing a file or directory in the TIFPS environment.

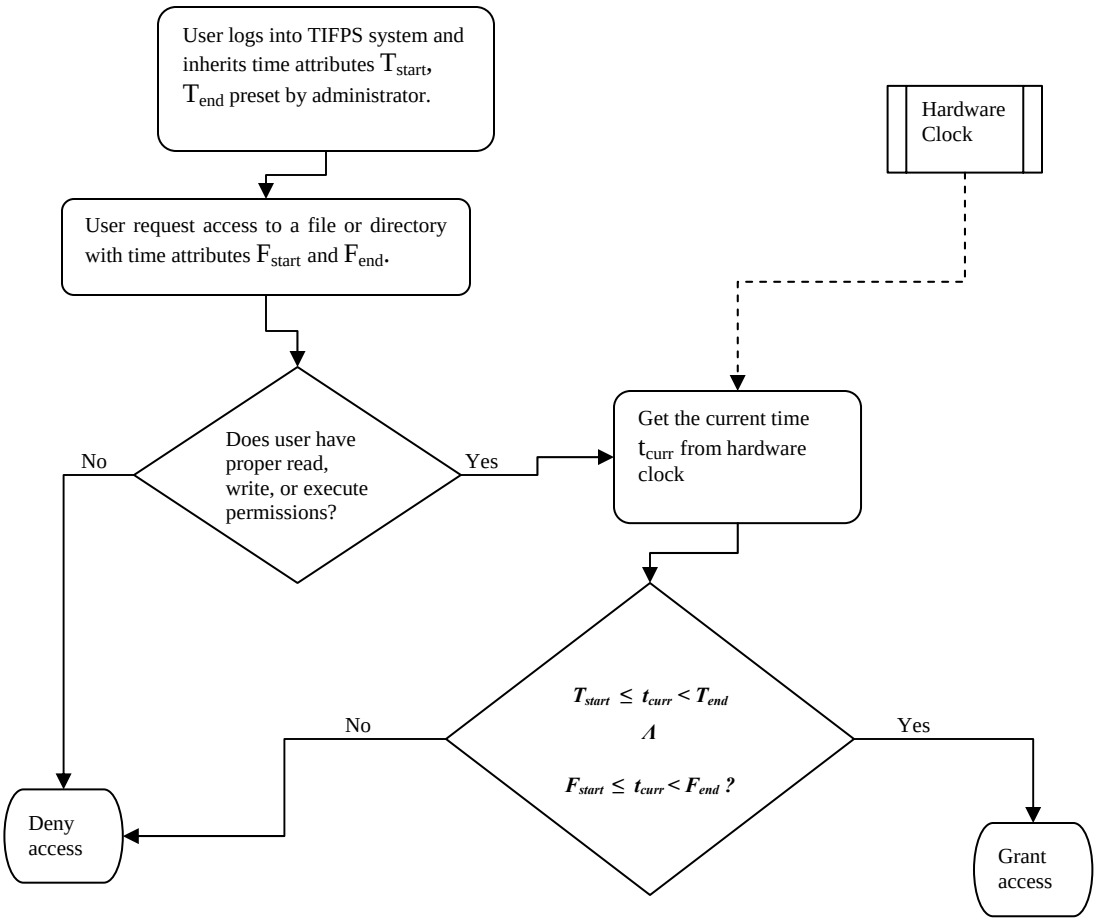


Figure 3-1. High level process flow for a user accessing a file or directory in TIFPS

When a user logs into the system, his login shell inherits the time interval attributes T_{start} and T_{end} specified ahead of time by a system administrator. T_{start} and T_{end} define the time interval of allowed access for the login shell process running on behalf of the user. Execution of programs within the shell will result in the program inheriting the time attributes from the user shell. When the program attempts to access a file, the system checks the current time t_{curr} against the time interval attributes T_{start} and T_{end} of the program inherited from the shell. If current time is within the time interval specified by T_{start} and T_{end}, then the file's time attributes are checked. F_{start} and F_{end} define the time interval of allowed access for the file or directory. The system administrator also specifies ahead of the time the time interval attributes F_{start} and F_{end} for the files and

directories that he wishes to control. If the current time falls within the time interval specified for the file or directory, then the system grants access to the file or directory (note that the standard Linux read, write, execute permissions remain in effect in addition to the time-based access control). Mathematically, an access is granted in TIFPS only if the following is true:

$$T_{start} \leq t_{curr} < T_{end} \quad \Lambda \quad F_{start} \leq t_{curr} < F_{end}$$

To prevent a user from extending the time-of-allowed access to the information in a file, which could occur if the user created a new file and copied the information from the time-checked file to the new one, the following access policy regarding the creation of file shall be implemented in the system. After a program reads in files with time interval attributes T1 and T2, any write operation to new or existing files will transfer the most restrictive time interval from all the files read to the files written. See Figure 3-2 for a diagram of the policy.

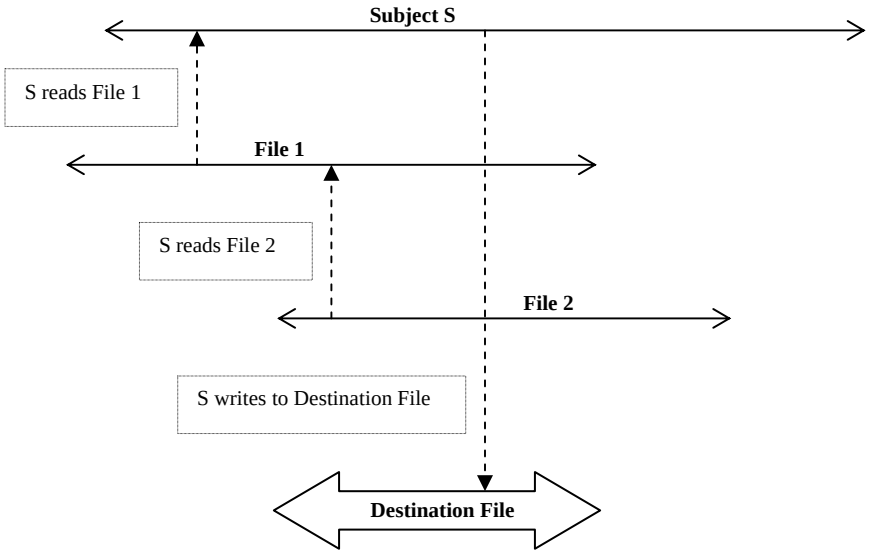


Figure 3-2. Diagram on TIFPS system read and write policy

Assume that the policy regarding creation of files above is not the case. When a program reads a program that is expires five minutes from now and subsequently writes the contents of that file into a second file (an effective copy operation), after the first file expires, the user will be able to continue reading the new file created as well as make new copies of that file, thus extending the time-of-allowed access to the contents in the original file.

2. Time Attribute Modification Tool High Level Description

Since the Linux 2.6 kernel series supports extended attributes for most Linux file systems, TIFPS will use extended attributes for specifying the time attributes. Fedora Core 5 as well as other Linux operating systems running Linux 2.6 and up include a set of user-space programs for setting and getting extended attributes, *setfattr()* and *getfattr()* respectively. The time attribute modification tool can be designed to utilize these existing tools. A wrapper program that packages these existing tools can be designed to set and modify the time attributes, get the time attributes, and present the time attributes, in a human understandable format. To meet the requirements described above, command line interfaces similar to standard Linux command line tools will be used to design the tool. Different flags can be used at the command line to set, delete, or display the time attributes of a file or directory. The "*" character can use used at the command line to specify multiple files. If incorrect flags are used, usage instructions will be displayed and the tool will exit without having an effect. A **man** page describing the usage of the tool will be available.

C. IMPLEMENTATION CHOICES

This section discusses the implementation choices made for the TIFPS kernel and the TIFPS tool and the rationale behind these decisions.

1. TIFPS Kernel Implementation Choices

Before starting the development effort on TIFPS, research into different security frameworks was done. Implementing and creating custom security hooks in the Linux Kernel specifically for TIFPS was considered but quickly abandoned given that a security framework with needed security hooks already exists. The Fedora Core 5 (FC5) distribution includes NSA's Security Enhanced Linux (SELinux), an access control mechanism, which uses the Linux Security Module (LSM). The LSM framework was

designed to be a modular security framework which provides security hooks called by the kernel in strategic locations in the kernel. For example, Linux's virtual file system calls *vfs_read()* and *vfs_write()* calls the LSM security hook *security_file_permisson()*. See Figures 3-3 and 3-4 for source code for *vfs_read()* and *vfs_write()*, respectively. The *security_file_permission()* function, along with other security hook functions are defined in the linux/include/security.h header file. These generic security hooks can be implemented to enforce different security policies and behaviors.

```
ssize_t vfs_read(struct file *file, char __user *buf, size_t count, loff_t *pos)
{
        ssize_t ret;

        if (!(file->f_mode & FMODE_READ))
                return -EBADF;
        if (!file->f_op || (!file->f_op->read && !file->f_op->aio_read))
                return -EINVAL;
        if (unlikely(!access_ok(VERIFY_WRITE, buf, count)))
                return -EFAULT;

        ret = rw_verify_area(READ, file, pos, count);
        if (ret >= 0) {
                count = ret;
                ret = security_file_permission (file, MAY_READ);
                if (!ret) {
                        if (file->f_op->read)
                                ret = file->f_op->read(file, buf, count, pos);
                        else
                                ret = do_sync_read(file, buf, count, pos);
                        if (ret > 0) {
                                fsnotify_access(file->f_dentry);
                                current->rchar += ret;
                        }
                        current->syscr++;
                }
        }

        return ret;
}
EXPORT_SYMBOL(vfs_read);
```

Figure 3-3. Source code for *vfs_read()* showing call to *security_file_permission()*

```
ssize_t vfs_write(struct file *file, const char __user *buf, size_t count, loff_t *pos)
{
ssize_t ret;

if (!(file->f_mode & FMODE_WRITE))
        return -EBADF;
if (!file->f_op || (!file->f_op->write && !file->f_op->aio_write))
        return -EINVAL;
if (unlikely(!access_ok(VERIFY_READ, buf, count)))
        return -EFAULT;

ret = rw_verify_area(WRITE, file, pos, count);
if (ret >= 0) {
        count = ret;
        ret = security_file_permission (file, MAY_WRITE);
        if (!ret) {
                if (file->f_op->write)
                        ret = file->f_op->write(file, buf, count, pos);
                else
                        ret = do_sync_write(file, buf, count, pos);
                if (ret > 0) {
                        fsnotify_modify(file->f_dentry);
                        current->wchar += ret;
                }
                current->syscw++;
        }
}
return ret;
}
EXPORT_SYMBOL(vfs_write);
```

Figure 3-4. Source code for *vfs_write()* showing call to *security_file_permission()*

One other Linux security framework was found during early phase research, it is called Rule Set Based Access Control (RSBAC) [4]. The author/maintainer of this framework suggests some weaknesses in the LSM. He mentions that LSM requires that the security hook functions be exported to user-space programs which make them vulnerable to root-kits. He also suggests that the set of security hooks is not complete. He speculates that the LSM support may be removed from the Linux Kernel in the future. The RSBAC framework project also cites another project's stance against using LSM, Grsecurity [5]. Grsecurity is a multi-layered, detection, prevention, and containment model for Linux security. Some of its features include kernel stack randomization, kernel null pointer dereference protection, and Role-Based Access Control.

Despite these arguments, it is still unclear whether LSM support will be removed from the Linux Kernel in the future and what its replacement might be. For the purpose of prototyping TIFPS, LSM was chosen as the framework for development. Since TIFPS

is only an access control system used for files and directories, only a subset of the security hooks provided by LSM will be sufficient to implement the system. By using the LSM, rapid prototyping TIFPS could be quick and efficient.

To use the Linux Security Module framework to build a loadable security module, the *__init()* and *__exit()* functions must be defined. The *security_operations* struct, which is a struct of function pointers for all of the security hooks, is used to implement custom security functions for each of the security hooks. For example, the *security_file_permission()* security hook is implemented by setting *.file_permission()* equal to *tifps_file_permission()* in the *security_operations* struct and by implementing the *tifps_file_permission()* function. When the kernel calls the *security_file_permission()* hook, the *tifps_file_permission()* will be called. It is sufficient to implement only the security hook functions necessary to achieve the desired system behavior. Any security hooks not defined will default to a set of dummy security hook functions defined in linux/security/dummy.c.

As suggested in the Chapter II, extended attributes for Linux files are provided in the 2.6 series Linux kernels. TIFPS will assign temporal attributes to files and directories using extended attributes. This means that a file system which supports extended attributes must be used. "Ext3" is a popular journaling file system that is installed by default and supports extended attributes. This prototype of TIFPS will assume the use of an "ext3" file systems. However, to the extent possible, the prototype shall be kept sufficiently generic to support other file systems that use extended attributes. For example, "ext2" and "xfs" are two other file systems that currently support extended attributes.

2. Time Attribute Modification Tool Implementation Choices

As mentioned earlier, there exist a set of tools for setting and getting the extended attributes for files, respectively setfattr and getfattr. Setfattr can only be run by the administrator account as described by the **man** pages, while getfattr can be run by any user to get the extended attributes of a file or directory. Since **bash** scripts are useful in running other existing command line programs and have support for parsing command line flags**, bash** scripts were chosen over other high level programming languages such as C and C++ for simplicity for developing the time attribute modification tool.

D. LOW LEVEL IMPLEMENTATION DETAILS

Low level implementation details of TIFPS LSM and the TIFPS tool are discussed in this section.

1. TIFPS LSM Low Level Implementation Details

TIFPS Security Data Structures

In the Linux kernel, a *task_struct* struct represents processes and an *inode* struct represents files, directories, and other file system objects. The Linux Security Module predefines in each of these data structures a security object pointer that points to a security struct custom defined by the specific LSM implementation. In this TIFPS LSM implementation, the security struct defined for processes is named *tifps_task_security_struct* and has the following fields: a 4-byte back pointer to the *task_struct*, a *semaphore* data structure used for synchornization, and two signed integers representing the start and end times of the time interval for allowed access by the process. The *inode* security struct is named *tifps_inode_security_struct* and has the following fields: 4-byte back pointer to the *inode* struct, a *semaphore* data structure, and two signed integers representing the start and end times of the time interval for allowed access to the file or directory object represented by the *inode* struct. See Appendix A, Section A, for the header file tifps_sec_objects.h defining these security data structures.

TIFPS Representation of Time

The notion of time in Linux is represented by a 4-byte signed integer, which specifies the number of seconds since the Unix epoch (January 1st, 1970 at 00:00:00 UTC). A negative integer represents the number of seconds before the Unix epoch. Since there is no practical benefit of specifying an allowed access time starting or ending prior to 1970, for simplicity, the TIFPS attribute has the range of 0x00000000 to 0x7FFFFFFF.

TIFPS Extended Attributes and String Format

The TIFPS security data structures described earlier are non-persistent representations of time attributes for processes, files, and directories in kernel memory. By non-persistent, it is meant that these data structures do not persist between hardware shutdowns. Extended attributes are used for persistent storage of the TIFPS security time

attributes and are stored as strings. The string representation of the name of the extended attribute for TIFPS is “security.tifps”. The value of the extended attribute has the format “:0x00000000:0x7FFFFFFF\0”, where the first hexadecimal number represents the start time of allowed access and the second hexadecimal number represents the end time of allowed access. Storing time attributes in this format using hexadecimal integer representation simplifies string parsing for manipulating these fields during access control operations.

TIFPS Enforcement Logic

The following is a description of how TIFPS enforces time-based access control policies. On system initialization, with TIFPS LSM loaded, the kernel allocates a *tifps_task_security_struct* for the current running process, initializes the *semaphore* struct, and sets the TIFPS start and end times to 0x00000000 and 0x7FFFFFFF, respectively. Subsequent tasks that are scheduled to run are also allocated a *tifps_task_security_struct*. Figure 3-5 below is a flow chart for the low level time policy enforcement logic.

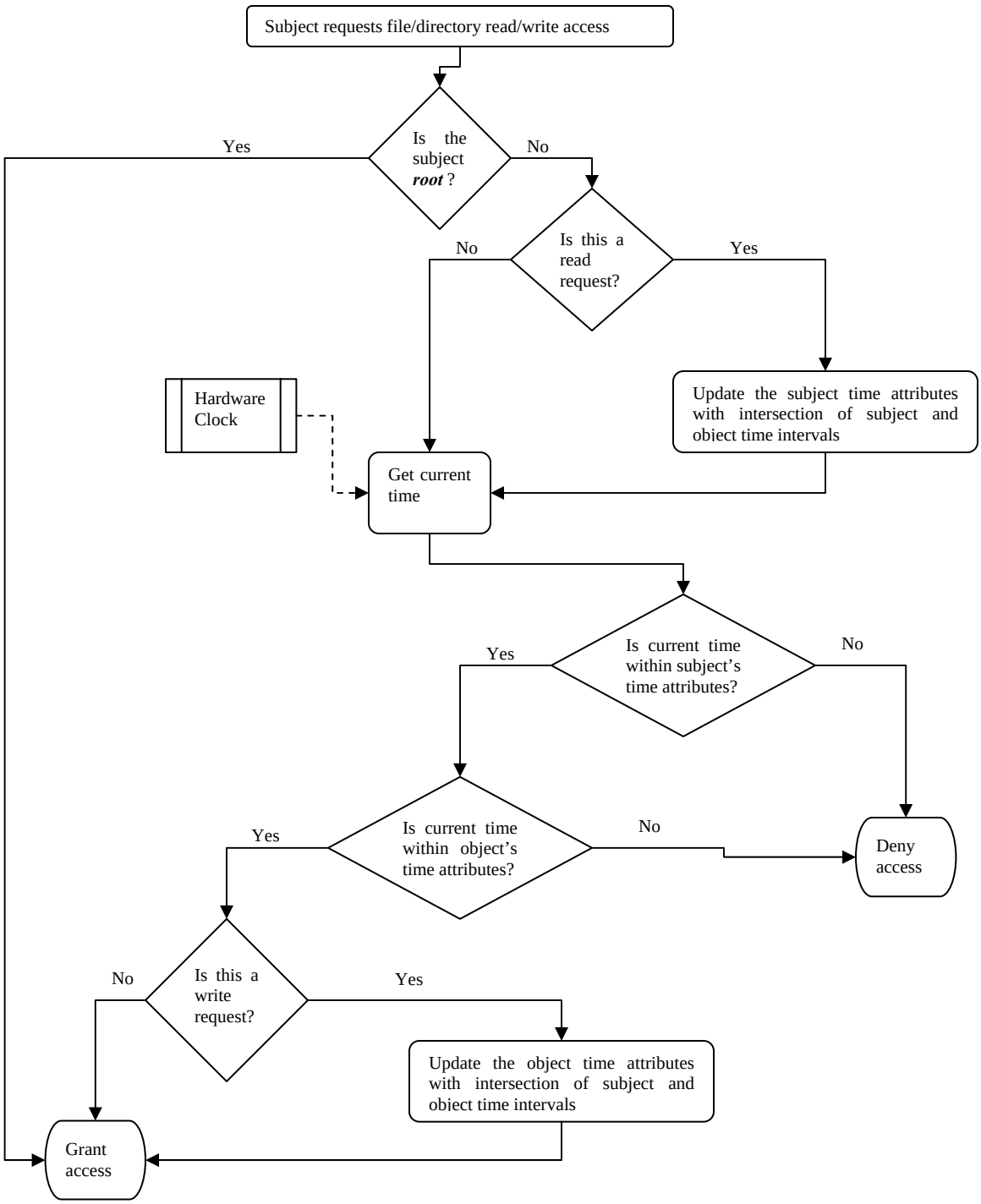


Figure 3-5. Flow chart for low level TIFPS enforcement logic

At every file/directory read and write access, the following checks take place. First check to determine whether the user represented by the process is ***root*** (i.e., has the *CAP_SYS_ADMIN* capability). If is the user is ***root***, then access is granted; otherwise, check that the current time falls within the time interval of the *tifps_task_security struct*. If not, access is denied, otherwise, check that the current time falls within the time interval of the *tifps_inode_security_struct.* If not, access is denied, if so, access is granted. To prevent unauthorized extension of access to information by copying, when read access to an object is requested in TIFPS, the process's time attributes are updated to take on the intersection of the time attributes of the object being read and the process's current time attributes. When write or append access to an object has been granted by TIFPS, the object's time attributes are updated to take on the intersection of time attributes of the requesting process and the object being written.

The reasons for updating the security structs after read and write operations are two-fold. The first reason is to prevent extension of access to information as described in Section B of this chapter. A second reason for the policy requiring a task to inherit the most restrictive attributes of files read is the notion of subject access control. The idea was presented in the TIAC model [1,3] where an administrator can grant and revoke time-based access to users in addition to controlling access to file and directory objects. In Linux, when a user logs in, the /etc/passwd file is read by the system to get the user's home directory. The user's login shell then changes the directory to the home directory specified. If an administrator sets the time attributes of the user's home directory, the user's time-of-allowed access to any files in the system besides his own home directory will be subject to the time attributes set for his home directory because of the task inheritance policy.

Preserving Time Attributes Across Copy Operations

The policy of continual restriction of the time interval for a process on object reads introduces a problem however. Assume that a user reads a file that expires in 5 minutes first after logging into the system. After reading the file, the process's time-of-

allowed access also expires in 5 minutes due to the inheritance. Therefore, after 5 minutes, the task will not be allowed to access any other files in the system, creating an undesirable condition for the user.

A modification of the policy was considered as described below in an attempt to address this issue but was not implemented, as will be explained. Since the system is intended for use in preserving the time attributes of file objects on copy operations, the *tifps_task_security_struct* can be implemented to "keep track of" (as opposed to inherit) the most restricted time attributes of files that it has read. Only during an attempt to write would the system enforce access control and transfer the time attribute with the most restrictive time interval to the file(s) being written. This solution was not implemented because the file read operation implies a write operation to the kernel stack. Also, an administrator's ability to grant and revoke time-based access control to users would not work in such a scheme.

Fortunately, the fork-and-exec paradigm of Unix-based operating systems solves the problematic condition. When a user logs into a Unix system, that user's login shell runs as a process. Any programs that the user decides to run from this shell causes the login shell to fork into a parent and child processes. It is the child process that executes the command, reads from, and writes to files. Because the parent login shell does not read or write files in the program execution, its time attributes assigned at user logon are preserved.

TIFPS LSM Security Hook Implementation Details

The TIFPS policy described above and the permission check logic for TIFPS are implemented in the *tifps_enforcer()* function in the helper functions section of the tifps_hooks.c source code file. The file is divided into two sections, one implementing the security hook functions called by the kernel as part of LSM and another implementing all the helper functions that the security hook functions call to provide the time based access control. See Table 3-1 for a list and description of the security hook functions implemented for TIFPS and Table 3-2 for a list and description of the helper functions. The source code for TIFPS can be found in Appendix A, Section A.

Table 3-1. List of LSM security hook functions implemented in TIFPS

Generic security hook	**TIFPS security hook implementation**	**Description**
security_inode_alloc()	tifps_inode_alloc_security()	Allocate and attach a TIFPS security structure to inode->i_security. The i_security field is initialized to NULL when the inode structure is allocated.
security_inode_free()	tifps_inode_free_security()	Deallocate the TIFPS inode security structure and set inode->i_security to NULL
security_inode_init()	tifps_inode_init_security()	Initializes inode->i_security structure with extended attributes of the file referenced by the inode. Note: as directed by the linux/include/security.h file, this hook function is expected to allocate memory for the name and value of the function parameters via kmalloc(). The caller is responsible for calling kfree() after using them.
security_inode_permission()	tifps_inode_permission()	Called by the existing Linux permission() function to additional permission checking.
security_inode_post_setxattr()	tifps_inode_post_setxattr()	Updates the inode security field after successful setxattr() operation.
security_inode_setsecurity()	tifps_inode_setsecurity()	Similar to *_inode_post_setxattr(), it is called by vfs_setxattr() if the file system does not support the setxattr() function.
security_file_permission()	tifps_file_permission()	Checks file permissions before accessing an open file on read and write operations.
security_task_alloc()	tifps_task_alloc_security()	Allocate and attach a security structure to the process's security field. The security field is initialized to NULL when the task structure is allocated.
security_task_free()	tifps_task_free_security()	Deallocate and clear the process's security field.

Table 3-2. List of TIFPS helper functions

TIFPS helper function	Description
tifps_time_to_xattr_value()	Converts a set of TIFPS start and end time attributes into the TIFPS format string to be stored as extended attributes
tifps_get_times()	Given a TIFPS-formatted string, parse the string to get the TIFPS start and end times.
tifps_helper_task_alloc_security()	The tifps_task_alloc_security() hook calls this function. It is defined as a helper function because it is also called by tifps_inode_permission() if a task does not have a security struct allocated yet.
tifps_update_task_security()	Updates the task TIFPS attributes with the intersection of the task and inode security structure time intervals.
tifps_update_inode_security()	Updates the TIFPS attributes for an inode with the intersection of the task and inode security structure time intervals.
tifps_enforcer()	The access control policy enforcer.
tifps_inode_has perm()	Checks with the enforcer as to whether access to an inode is allowed. This function is called by the security hook tifps_inode_permission() during initial opening of files. It is also called by tifps_file_has_perm() for ongoing file descriptor access.
tifps_file_has_perm()	Checks with the enforcer on whether ongoing access to a file is permitted.

TIFPS LSM Configuration, Compilation, and Installation

As the name Linux Security Module suggests, TIFPS was designed as a loadable module for the Linux Kernel. However, the kernel configuration utilities have been modified to compile TIFPS as either a loadable module or as a module permanently built into the kernel. See Appendix A, Section A for copies of the Kconfig and Makefiles edited for this purpose and Appendix B, Section A for screenshots of the kernel configuration menu for TIFPS LSM included as part of the installation procedures. Please note that since this project was a proof of concept prototype, compatibility with other security modules such as NSA's SELinux or BSD's Secure Level LSM has not been considered or tested. BSD's Secure Level LSM provides increasingly restrictive levels of security. All testing of TIFPS functionality has been done without compiling SELinux or any other non-traditional Linux security modules support.

2. Time Attribute Modification Tool Usage and Implementation Details

The **modtime** command line tool is the time attribute modification tool written using a combination of the **bash** scripting language and **perl.** It is used to convert a TIFPS string stored as extended attributes on a file to a date and time that is easily interpreted by a human user. The tool is intended for use by administrators to set and

modify TIFPS attributes for files and directories. Through the use of flags, the tool can also be used by users and administrators to view the TIFPS attributes in human readable format.

When the program is executed, the number of arguments is checked, if no arguments are given, a usage instruction is given. The usage format for the tool is given below:

modtime <flags and corresponding flag arguments> <files and/or directories>

Note that multiple files and directories can be given to the tool.

The program uses the *getopts* built-in command tool for **bash** to parse flags given on the command line. There are three modes of operation for the **modtime** tool: get time attributes, set time attributes using absolution time, and set time attributes using relative time (relative to current time). As a user, the –g flag can be given at the command line to get time attribute information about file and directories. As an administrator, **modtime** can be used to set time attributes. The –a and –A flags are used to set the absolute start and end times, respectively. The argument following the flag must be a string recognizable by the **date** command in Linux. For example, the command:

 # modtime –a now –A "9/22/06 17:00:00EST" myfile.txt

sets the time attributes for *myfile.txt* to allow access starting now and to revoke access on 9/22/06 17:00:00 Eastern Standard Time. Note the Linux time system automatically accounts for time zones and converts the time zone specified to the time zone configured for the system. Sample output:

Target: myfile.txt

Grant access on: Sun Aug 13 16:50:52 2006

Revoke access on: Fri Sep 22 15:00:00 2006

To set the time attributes relative to current time, the following flags are used:

-s, -S, -m, -M, -h, -H, -d, -D, -w, -W. The lower case flags correspond to relative start times while the upper case flags correspond to relative end times for the target. "sS" flags set the time seconds from now; "mM" flags set the time minutes from now; "hH"

flags set the time hours from now; “dD” flags set the time days from now; and “wW” flags set the time weeks from now. Negative integer arguments following the relative time flags indicate an earlier time from the current time while positive integer arguments indicate later times relative to the current time. For example, the following command:

 # modtime –s -30 –m 5 –W5 /home/user

sets the start time attribute for the directory */home/user* to 4 minutes 30 seconds from now (5 minutes minus 30 seconds) and the end time attribute to five weeks from now.

Table below shows a summary of the command line flags and their intended use. Appendix A, Section B contains the source code for the tool.

Table 3-3. Summary of **modtime** command line flags and its usage

Flags	**Description of Usage**
-g	Displays the time attributes of the file or directory
-x	Deletes the time attributes of the file or directory
-a	Sets the absolute time for allowing access to the file or directory
-A	Sets the absolute time for revoking access to the file or directory
-s	Sets the relative time to current time in seconds for granting access to the file or directory
-S	Sets the relative time to current time in seconds for revoking access to the file or directory
-m	Sets the relative time to current time in minutes for granting access to the file or directory
-M	Sets the relative time to current time in minutes for revoking access to the file or directory
-h	Sets the relative time to current time in hours for granting access to the file or directory
-H	Sets the relative time to current time in hours for revoking access to the file or directory
-d	Sets the relative time to current time in days for granting access to the file or directory
-D	Sets the relative time to current time in days for revoking access to the file or directory
-w	Sets the relative time to current time in weeks for granting access to the file or directory
-W	Sets the relative time to current time in weeks for revoking access to the file or directory

E. DEVELOPMENT ENVIRONMENT

In addition to LSM, the following tools were used to facilitate development.

1. VMware Server 1.0.0

VMware Server is free virtualization software that virtualizes hardware for running different operating systems on the same hardware. It was used for the development of TIFPS both to run a dedicated Subversion versioning server and the test kernel where the development and testing took place. A 20 gigabyte VMware image was created for the Subversion server and 10 gigabyte VMware® images were created for development and testing purposes.

2. Subversion 1.3.0-4.2

Subversion is an open-source versioning software used to control versions of documents and source code being modified from different machines [6]. It allowed the flexibility of development from multiple workstations. It also provided a critical backup of the entire development project. Daily commits to the Subversion server guaranteed that there will always be two copies of the latest work in the event that an unforeseen disaster strikes.

3. Source Insight 3.5

Source Insight is a source-code visualization software [7]. It creates function call graphs for quick visualization of the overall code structure. It also provides convenient browsing of the code providing links to functions variables, macros, and structures. Going from one function to another was as easy as double clicking the function name in the source. It was used to visualize and understand the existing source-code for kernel version 2.6.15.

4. Fedora Core 5 – Kernel 2.6.15

The Fedora Core 5 Linux distribution [8] with kernel version 2.6.15 was used as the target operating system for development as well as running the Subversion server. To minimize the time it took for a compile and test cycle, the minimum number of modules required to run the system was selected for kernel installation. Also, only absolutely necessary kernel drivers were compiled into the kernel. This also reduced build time. See Appendix E for a copy of the kernel configuration file.

5. Emacs 21.4-14

For modifying kernel source code in the developmental VMware® images, the **emacs** editor was used. Since the Linux kernel source code can be edited by any one and without coding standards, there is a significant potential for "messy" code. The Linux kernel source contains a "CodingStyle" document in the **linux/Documentation** directory. It specifies the conventions that anyone developing the kernel should follow. Specific guidance on indents, long lines, braces, naming, etc are given. An **emacs** configuration file that conforms to the coding style recommendations for indentations can be found in Appendix E, Section B.

F. SUMMARY

This chapter described the design and implementation details of the TIFPS LSM as well as of the **modtime** command line tool used for interacting with the system. Requirements for both the LSM and the tool were captured as part of the description. Implementation choices made during development were then discussed and rationale provided for these choices. Finally, the development environment used was presented. In the next chapter, testing of the TIFPS LSM and the analysis of test results will be discussed in detail.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TESTING AND ANALYSIS

This chapter describes test plans and analyses for validating TIFPS for correct functionality, measuring its performance overhead, and gauging its robustness in multi-user situations. To test the TIFPS Linux Security Module (LSM), the following major steps were followed in the testing process:

- Develop test plan
- Conduct tests
- Analyze results
- Correct system behavior, as needed, and retest

The results captured in this chapter reflect the final iteration of testing and include any modifications to the system during the iterative testing phase.

The test plan is divided into three categories described below. Functional tests of the access control mechanism test for proper enforcement of the time-based access policies. Performance testing to quantify the overhead of the added time-based access control of TIFPS LSM compared with an unmodified kernel when reading, writing, and copying of files. Finally, concurrency testing provides a gauge of the robustness of the TIFPS LSM in multi-user situations where attempts to access files and directories are concurrently made by different users.

A. ACCESS CONTROL TESTS

Access control tests were conducted to determine if the TIFPS LSM enforced the access control policies as expected. As a result of this testing, four unexpected problems related to the preservation of time attributes were encountered and discussed in Section A.3, the Analysis of Results section. Two of the four problems had simple solutions and were therefore fixed while only potential solutions are discussed for the remaining two.

The test plan for access control enforcements are described in Section A.1. Test results are reported in Section A.2. As mentioned before, the results reported include any attempt to fix the problem encountered in Section A.3 and do not reflect the iterations of testing that occurred. Section A.3 also discusses potential solutions for the expected

problem of access revocation during file writes. Before we begin describing the access control test plan, the general TIFPS access control and inheritance policies can be informally stated as follows:

- Access to an object shall be granted only if the current time is within the time interval defined by the intersection of the subject and object.
- Time attributes of copied files must be inherited from the intersection of time attributes for the subject, source object, and destination object.

1. Access Control Test Plan

The access control test plan is divided into two categories: static and dynamic tests. The static tests category includes test cases where the subject and object time attributes are preset by the administrator and remain unchanged during the tests. The set of static tests are further divided into the following sub-categories:

- Enforcement of time-based policies for reading, writing, and executing files and directories (executing files refers to the execution of binary executables, executing directories refers to changing into the directory)
- Inheritance of time attributes in file and directory creation operations and in file-copy operations
- Behavior of TIFPS when access time expires and access is revoked during file write operations

The third test sub-category from above is planned in anticipation for the potential problem of file corruption in the event file format information is incompletely written to a file due to access revocation. Directory writes are not considered because directory writes are atomic with respect to access checks therefore the same problem is not anticipated.

The dynamic tests category, on the other hand, covers the cases where the administrator changes the time attributes of subjects or objects while the user is logged into the system.

Static tests – enforcement of file and directory read/write/execute

The TIAC model [1,3] uses interval algebra to describe the temporal relationships between subjects and objects. Table 4-1 shows all the possible relationships between a subject and an object and the expected access permission in TIFPS. The first set of static tests was to determine if permission enforcement in TIFPS is consistent with the time-

based access control policy for each of these relationships. When a subject, S, attempts to access an object, O, access is only allowed during the period in which their time intervals of allowed access overlap. For example, in scenario 1, S has a time interval of allowed access specified by t2. The time interval of allowed access for O is specified by t4. The other time intervals t1, t3, and t5 specify periods where neither access to S or O is allowed. Given this scenario, access should be denied for all time intervals t1 through t5.

The objective of this set of tests is to check for proper enforcement of time based access control on read, write, and execution of files and directories at all time intervals given a set of subject and object time attributes related as shown in Table 4-1. In this portion of the static access control tests, each subject/object relationship in Table 4-1 was setup using **bash** scripts. Read, write, and execute operations are then performed on specified files and directories within each of the identified time intervals and the system behavior was verified with expected result.

Since the system should grant permission only when the time intervals of subjects and objects overlap, it is inferred that if the subject and object in Table 4-1 were swapped, the same access permissions will be expected. Rather than duplicating the entire test matrix of 42 (3 x 2 x 7) test cases for read, write, and execute operations on files and directories in each of the seven scenarios, which would be highly redundant, two test cases are to be selected semi-randomly and verified that the system grants proper permissions for all time intervals t1 through t5. The semi-randomly selected test cases shall have expected behaviors of both grant and deny access.

Table 4-1. Basic temporal interval relationships between a subject S and object O*

Test ID	Scenario	Relation	Pictorial Meaning and Access Intervals	Expected Access permission
A1	1	S before O O after S	< ----S------ > < -----O----- > t1 t2 t3 t4 t5	t1 to t5: deny
A2	2	S equals O O equals S	< ----------S---------- > < ----------O---------- > t1 t2 t3	t1 and t3: deny t2 : allow
A3	3	S meets O O met by S	< -------S-------><------------O------- > t1 t2 t3 t4 t5	t1 to t5: deny
A4	4	S overlaps O O overlapped by S	< ----------S---------- > < ----------O---------- > t1 t2 t3 t4 t5	t1, t2, t4, t5: deny t3: allow
A5	5	S during O O includes S	< ----------S---------- > < ----------O---------- > t1 t2 t3 t4 t5	t1, t2, t4, t5: deny t3: allow
A6	6	S starts O O started by S	< ----------S---------- > < ----------O---------- > t1 t2 t3 t4	t1, t3, t4: deny t2: allow
A7	7	S finishes O O finished by S	< --------S------- > < ----------O---------- > t1 t2 t3 t4	t1, t2, t4: deny t3: allow

*Note: the access permissions would be the same if S and O were swapped.

Static tests- Inheritance in file/directory creation and file copy operations

The objective of this set of tests is to check for proper preservation of time attributes during file and directory creation and file copy operations. In this set of tests, time attributes of files and directories were displayed after creation by a user whose time attributes (represented by the subject time attributes) had been preset by the administrator. Expected behavior is that the files and directories created will inherit the time attributes of the user. Table 4-2 below summarizes the two test cases: one with a user creating a new file and another with a user creating a new directory.

Table 4-2. File and directory creation tests and expected results

Test ID	Test case	Expected Result
B1	User creates new file	The new file should inherit the time attributes of the user.
B2	User creates new directory	The new directory should inherit the time attributes of the user.

In the set of copy tests, three scenarios of a user subject copying content from a source file to a destination file were envisioned. In the scenarios, the subject, source object, and destination object each have different time attribute relationships as depicted in the Figure 4-1 below. It is expected that the destination object will inherit the time attributes of the intersection of the three entities involved. The expected inherited time interval for the created file is illustrated in the figure. For each of the three scenarios, three ways to copy files in Linux are to be tested:

- Using the **cp** command
- Using redirection '>'
- Using pipes '|'

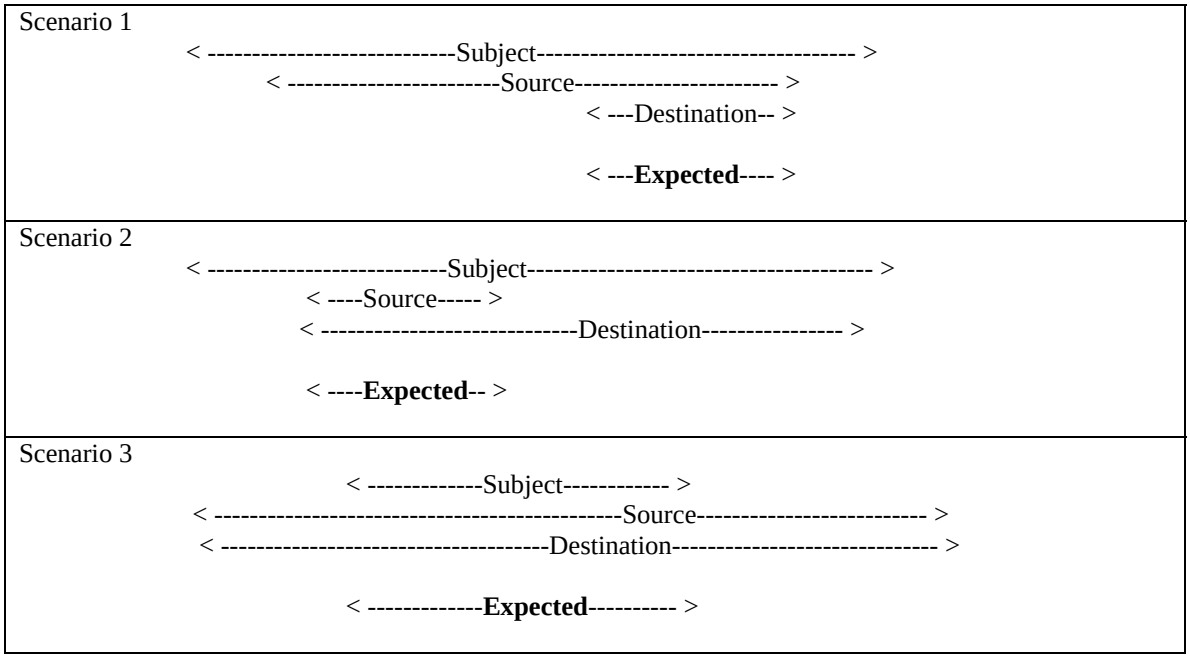


Figure 4-1. File Copy Scenarios.

Each test case in the 3x3 matrix will be performed 10 times to check for consistent behavior, see Table 4-3.

Table 4-3. Time attribute Inheritance on File Copy Test Matrix

Test ID	Scenario	Copy Method		
		cp	**Redirection '>>'**	**Pipes '|'**
C1 – C3	1	10 trials	10 trials	10 trials
C4 – C6	2	10 trials	10 trials	10 trials
C7 – C 9	3	10 trials	10 trials	10 trials

Static tests– TIFPS behavior on time expiration during file-write operations

The objective for this set of tests is to observe the behavior of the system when access to a file is revoked during a write operation. It is speculated that file corruption will occur if access to a file expires while an application is writing state or format information to the file. As such, the tests attempt to write a large amount of information (5M bytes) to files whose expiration time does not allow for the completion of the write operation. For convenience, a **bash** script is setup to take the expiration time of the file to be written-to as an argument. Immediately after setting the time, the script attempts to write 5 million bytes of information to the file. Next, the script counts the number of characters written to the file successfully. The script will be executed in multiple runs. For each run, the time-to-expiration (TTE) is increased. The test is complete when a TTE allows all 5 million bytes of information to be written successfully (TTE-max). It is expected that prior to reaching TTE-max, only part of the 5 million bytes of information will be successfully written to the file. Error messages that occur during each run will be captured for discussion. Table 4-4 shows the information to be captured for this test set.

Table 4-4. Sample table for information to be captured for the access revocation during file write tests

Test ID	Time to expiration (TTE)	Number of bytes written successfully out of 5 million	Error Message
D1	*Record expiration time used*	*Record # of bytes written*	*Record kernel error message here.*

Dynamic tests – Dynamically changing subject and object attributes

The objective of these tests is to observe the behavior of the system when time attributes are dynamically changed by an administrator while a user is logged in. A main

bash script was setup to initialize a pair of subject and object entities by setting their respective time attributes using an administrator account. After the initialization, the main script sleeps long enough to allow a human tester to run a second script as the subject (user script). The user script is setup to read the object before and after a time attribute change by the administrator. Next, the main script wakes from sleep and changes the time attributes of the subject or object. See Figure 4-2 for an illustration of the progression of these tests. System behavior from the subject (user)'s perspective is recorded before and after the change by the administrator.

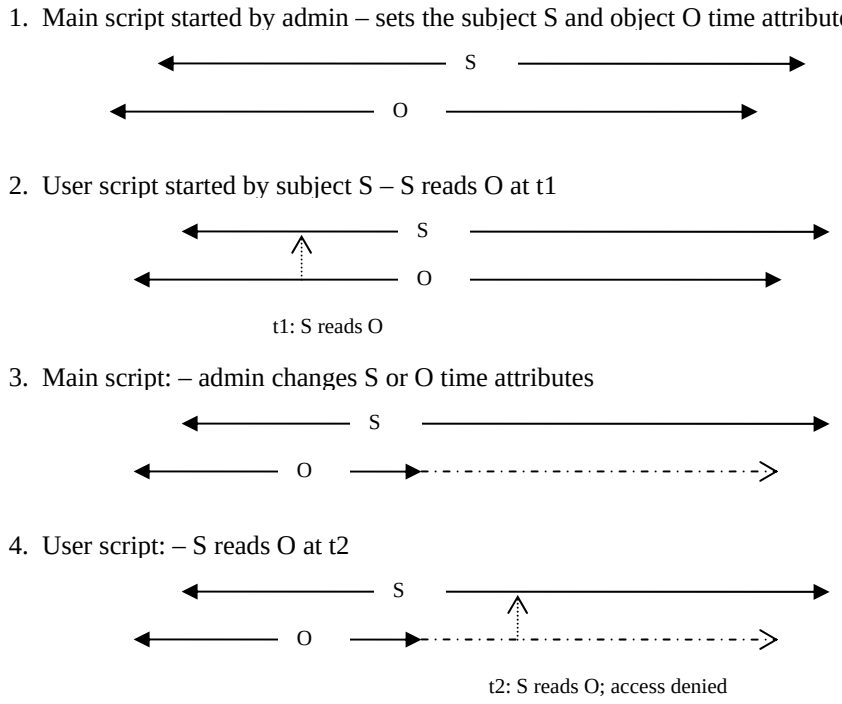


Figure 4-2. Dynamic Test Progression Illustration

There will be two test cases, one in which the administrator changes the time attributes of the subject and the other the temporal attributes of the object are modified. The expected results are summarized in Table 4-5 below.

Table 4-5. Summary of expectations for dynamically changing subject and object time

Test ID	Test Case	Expected Results
E1	Change subject time	Continued access should be allowed since time attributes are inherited at user login.
E2	Change object time	Access should be revoked according to the newly set time attributes.

2. Results

As mentioned earlier, the results shared here include all modifications to the system when it was necessary to address the unexpected problems discussed in Section A.3. These results do not reflect the iterations that occurred between modifications.

Static tests results

Table 4-6 is a summary of the results from the static tests for file and directory read, write, and execute permission enforcement. These tests resulted in expected behavior for all test scenarios. The test scripts and screen captures for each individual test can be found in Appendix D.

Table 4-6. Results from static tests for file and directory read/write/execute

Test ID	Scenario	Files				Directories		
		Read	Write	Exec		Read	Write	Exec
1	1	Pass	Pass	Pass		Pass	Pass	Pass
2	2	Pass	Pass	Pass		Pass	Pass	Pass
3	3	Pass	Pass	Pass		Pass	Pass	Pass
4	4	Pass	Pass	Pass		Pass	Pass *	Pass
5	5	Pass	Pass	Pass		Pass	Pass	Pass
6	6	Pass *	Pass	Pass		Pass	Pass	Pass
7	7	Pass	Pass	Pass		Pass	Pass	Pass

* Note: the asterisk indicate additional testing where subject and object were swapped for the test case and the results which were also found to be successful.

The static tests for file and directory creation resulted in expected time attribute inheritance behavior, details can be found in Appendix D, Section C.

For copy inheritance, the test results were as expected except for the test set using pipes. Table 4-7 summarizes these results. Test scripts and results for each individual scenario can be found in Appendix D, Section B and Section C, respectively.

Table 4-7. Summary of results for static tests for file copy operations

Test ID	Scenario	'cp'	Redirection '>'	Pipe '|' to 'tee'
C1 – C3	1	10 out of 10 pass	10 out of 10 pass	10 out of 10 pass
C4 – C6	2	10 out of 10 pass	10 out of 10 pass	**9 out of 10 pass**
C7 – C9	3	10 out of 10 pass	10 out of 10 pass	10 out of 10 pass

Table 4-8 summarizes the test results for access revocation during file write operations. The results confirm our speculation that file corruption could occur if access is revoked while an application is writing state information to a file. The resulting error messages when access permissions were revoked at different times during a write operation are also captured.

Table 4-8. Summary of results for access revocation during file writes

Test ID	Time to expiration (TTE)	Number of bytes written successfully out of 50 million	Error Message
D1	1	0	ERROR opening file
	2	49,152	ERROR writing to file: ERR -1
	3	2,002,944	ERROR writing to file: ERR -1
	4	3,338,240	ERROR writing to file: ERR -1
	5	5,000,000	None

* Note: The extra byte written to the file is a carriage return

Dynamic test results

As expected, dynamically changing the subject's time attributes does not affect a user's continued access to files and directories in this implementation of TIFPS. This was expected because the user inherits time attributes at the time of login. Since file and directory read and write operations are checked at every access request, dynamically changing the attributes of the objects in the system results in successful revocation of the object upon expiration of the object's temporal access. This test also produced the expected results. See Table 4-9 below for a summary of results.

Table 4-9. Summary results for dynamically changing subject and object time attributes

Test ID	Test Case	Results
E1	Change subject time	Continued access allowed.
E2	Change object time	Access revoked according to the new time attributes.

3. Analysis of Results

During testing, four unexpected problems with the TIFPS implementation were encountered. The first two discussed were fixed while the remaining two were analyzed for potential solutions. The anticipated problem of access revocation during file write is also analyzed and discussed in this section.

Directories inheriting task attributes restricting user access to files

First, a user's access to files in his or her home directories became increasingly restrictive as he copies files with more restrictive attributes. Since directories were implemented to inherit time attributes just as regular files do, files with less restrictive time attributes in a modified directory will not be accessible to the user. Also, as a user reads from directories, the task data structure associated with his login shell inherited the more restrictive attributes, preventing further access to other files in the system that he might otherwise be allowed to access. This is even more problematic when a directory, i.e. **/tmp** is shared among different users because one user can prevent access of other users sharing the directory. The problem was observed in the static tests for proper attribute inheritance in file copy operations.

The fix to this problem was to simply ignore time attribute updates on all directory-related operations in the TIFPS implementation. The results reported in the previous section include this change.

Inconsistent inheritance of task attributes

A problem of incorrect inheritance of time attributes for processes after reading files was observed. This problematic behavior of TIFPS was caused by our incorrect assumption of when the LSM security hook, *security_task_alloc()* function is called. It was assumed that *securiy_task_alloc()* is called after forking was complete. Actually,

this security hook function is called from the *copy_process()* kernel function which clones the parent before the cloned process becomes the forked child when the user login shell forks. For this reason, the forked child's parent was actually the parent of the login shell**,** rather than the login shell itself. In other words, the problematic implementation used the grandparent of the forked child rather than the parent to determine the time attributes of the forked child. The solution was simply to use the process being copied to determine the child process's time attributes. The results reported in the previous section also include this fix. This problem also occurred in the static tests for proper inheritance in file copy operations.

The piping problem

Next, in an effort to ensure that the system consistently enforced the inheritance policy for copying files, multiple ways of copying files in a Linux system were tested. The system behaved as expected except when pipes were used to copy files. The program **tee** reads from input and splits the bytes read from input into two streams. The first stream is written to standard out and the second stream written to a specified destination file. It can be used to copy file as in the following command:

$ cat source.txt |tee destination.txt

Since **tee** is reading from the pipe and the pipe does not have time attributes, this command successfully copies the contents of source.txt into destination.txt without preserving the time attributes of the source.txt file. Figure 4-3 below shows the relationship of the processes involved in the command above.

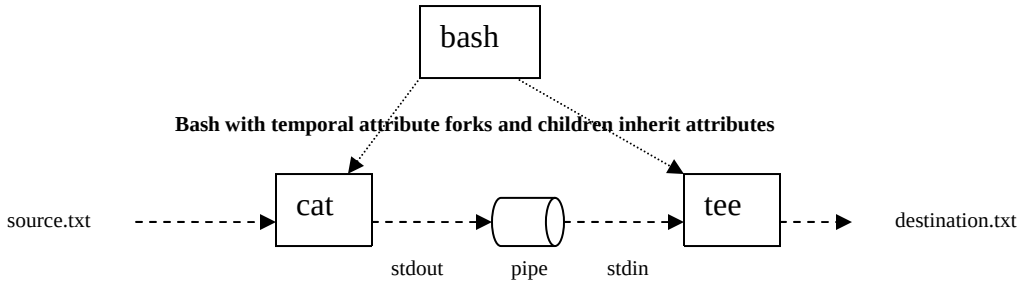


Figure 4-3. Using **tee** to copy files

An attempt was made to fix this by implementing time attribute inheritance for pipes. The results reported in the previous section include this implementation. In Linux, pipes are implemented with many of the properties of files and have inode data structures associated with them. Thus, they can be assigned time attributes just as regular files. The copy command above can be separated into the following individual operations. The actions in the parenthesis indicate envisioned TIFPS behavior for pipe attribute inheritance:

1. **cat** reads from **source.txt** (**cat** inherits attributes from **source.txt**)
2. **cat** writes to the pipe (pipe inherits attributes from **cat**)
3. **tee** reads from the pipe (**tee** inherits attributes from pipe)
4. **tee** writes to **destination.txt** (**destination.txt** inherits attributes from **tee**)

It can be seen that the **source.txt** time attributes are inherited through the chain of read and write operations by the **destination.txt** file. In theory, this suggests that implementing time attribute inheritance for pipes should fix the problem. However, the results reported in the results section indicate that the destination file is not inheriting the source file time attributes on a consistent basis. Upon closer inspection, the pipe copy command above does not necessarily occur in the order indicated in steps 1 through 4. The kernel scheduler was observed to schedule step 3 first for example, and the **tee** process will block until the **cat** process writes data to the pipe. Since the LSM security hook is called when the **tee** process requests read permission to the pipe and not after it wakes from blocking when data is written to the pipe, the time attributes of the original file will not be correctly inherited.

A potential solution is to change the security hooks for LSM in the kernel by enforcing a permission check after processes wake from blocks. This potential solution is outside of the scope of this thesis and has not been implemented.

Problem associated with assigning time attributes to executables in **bash**

The **bash** shell has a convenient tab-completion feature that allows a user to list all executables available in his/her path. When this feature is used, all the executables in a user's path are read by the login shell **bash.** Therefore, using this feature results in

bash inheriting the most restrictive time attributes of all executables in his/her path. For example, if the **/usr/bin/cal** program has been assigned by the administrator to expire in 5 minutes, any user logged in using the tab-completion feature will be effectively locked out of the system after 5 minutes. The users can logout of the system and re-login to circumvent this problem. This problem occurs only in login shells that have this auto-completion feature. Other shells, such as **ksh, tclsh, and tcsh** which do not have this feature do not exhibit this problem. This problem has not been fixed in this implementation and will be left for future work. It is recommended that in the meantime, time attributes only be set on non-executable files when using **bash**.

Incomplete write operations in the revocation of access during file writes

Finally, the TIFPS LSM does not provide transactional support for file writes. It is anticipated that this will be a problem when access to a file expires during the write operation. If important file state information has not been written to the file before the expiration, the file could potentially be left in an inconsistent state. Table 4-8 in Section A.2 shows two distinct error messages depending on when the access to the file is revoked. The error message "ERROR opening file" indicates that there was not enough time for the process to open the file for writing and therefore 0 bytes were successfully written. The error message "ERROR writing to file: ERR -1" indicates that the file had been successfully opened for writing but access was revoked when the process requested write permission to the file. Error number -1 is the number return by the kernel to indicate a permission-denied error. From these results, we confirmed that file corruption could potentially occur on write operations. One way to resolve this problem is to provide transactional support for the file system in the kernel. By providing a way to roll-back changes to the file, the system can keep the files in consistent states even if write operations fail due to revocation. The applications can also be designed to provide such support by keeping the state of the last successful write operation and reverting back to that state if new write operations fail. The file system tested in this prototype is "ext3" which supports journaling for quick file system recovery in the event of power failures and hardware failures. It is suggested that the journaling features of file systems such as "ext3" be investigated further as they may offer a potential solution to this problem.

B. PERFORMANCE TESTS

The objective of performance testing is to measure the additional overhead for doing time-based access checks by the TIFPS LSM compared with an unmodified kernel. The following sections describe the test plan, results, and an analysis of the results of performance testing. Overall, the added overhead for TIFPS access control is approximately 5% for read operations, approximately 20% for write operations, and approximately 9% for copy operations.

1. Performance Test Plan

A set of simple bash scripts were created to time the reading, writing, and copying of files on an unmodified 2.6.15 kernel and a kernel loaded with the TIFPS LSM. Comparisons between the two kernels were performed on a machine running virtualized VMware® server images of Fedora Core 5. The hardware running the VMware® image has an Intel® Pentium® 4 processor running at 3.00 GHz. The RAM allocated for the image is 256M.

In each of the three categories of read, write, and copy operations, some additional variables that were speculated to affect performance were also studied. First, the existence of time attributes on files may have an impact on the performance since TIFPS skips the logic for time-based access control check if an object does not have TIFPS attributes. Secondly, performing an operation on a single file 1000 times versus on 1000 different files once could affect the performance because more security data structures need to be allocated and initialized for the case where different files are handled. To study these two factors, four sets of tests were performed in each of the three categories. These are listed below.

- File operation (read/write/copy) on a single file 1000 times with existing TIFPS attributes.
- File operation on a single file 1000 times without TIFPS attributes.
- File operation on 1000 different files once; each file has existing TIFPS attributes.
- File operation on 1000 different files once; none of the file have existing TIFPS attributes.

Table 4-10 shows the general commands and tools used in bash scripts for each of the three categories (read, write, execute) in the performance test. The **time** tool was

used to record the time to run each script. Only system time is captured since access control occurs in the kernel. Refer to Appendix D, Section E for the actual test scripts used for performance testing. Table 4-11 is a summary of the descriptions of each test in the performance tests.

Table 4-10. Linux Commands and Tools used for Testing.

File Operation	**Linux Command**
Read	cat file(s).txt >/dev/null
Write	python –c "print 'G'*1000" > file(s).txt
Copy	Cp source-file(s).txt destination-file(s).txt

Table 4-11. Summary of description for the performance evaluation

Test ID	**Performance test variable descriptions**
F1	Read a single file with TIFPS attributes 1000 times
F2	Read a single file without TIFPS attributes 1000 times
F3	Read 1000 files with TIFPS attributes 1 time
F4	Read 1000 files without TIFPS attributes 1 time
F5	Write a single file with TIFPS attributes 1000 times
F6	Write a single file without TIFPS attributes 1000 times
F7	Write 1000 files with TIFPS attributes 1 time
F8	Write 1000 files without TIFPS attributes 1 time
F9	Copy 1 file with TIFPS attributes 1000 times to another existing file with TIFPS attributes
F10	Copy 1 file without TIFPS attributes 1000 times to another non existent file
F11	Copy 1000 different files, each with TIFPS attributes to another set of 1000 files, with TIFPS attributes
F12	Copy 1000 different files, without TIFPS attributes to a set of non existent files

2. Results and Analysis

A summary of the performance results is shown in Tables 4-12.

Table 4-12. Summary of performance for the 3.0Ghz Dell Desktop PC VMware® image*

	Read				Write				Copy			
	Single-file		Multi-file		Single-file		Multi-file		Single-file		Multi-file	
Kernel	**Attr**	**None**	**Attr**	**None**	**Attr**	**None**	**Attr**	**None**	**Attr**	**None**	**Attr**	**None**
Normal - avg	4.41	4.39	4.47	4.4	26.77	26.56	27.58	27.05	6.5	6.42	6.71	6.85
Normal – stdev	0.03	0.02	0.01	0.02	0.38	0.16	0.10	0.16	0.05	0.04	0.07	0.04
TIFPS - avg	4.65	4.59	4.72	4.65	32.28	31.91	32.59	32.2	7.09	7.09	7.25	7.4
TIFPS - stdev	0.03	0.03	0.03	0.02	0.41	0.22	0.52	0.28	0.07	0.02	0.10	0.03
Difference	5.44%	4.55%	5.51%	5.68%	20.6%	20.1%	18.16%	19.06%	9.13%	10.44%	8.05%	7.98%

*Note: Units are seconds unless otherwise noted

The results suggest that the presence of TIFPS attributes did not significantly affect the performance contrary to hypothesis. The reason for this result could be that most of the performance overhead of TIFPS occurs in the setup of the function calls to the TIFPS security hook implementations. In the TIFPS security hook implementations, access control logic is skipped in the absence of TIFPS attributes. It appears that skipping sections of code within a security hook function call did not significantly reduce performance overhead.

Also, with regard to comparison between multiple reads and writes to a single file and single reads and writes to multiple files, the results suggest that performing single-file operations does not have significant performance advantages over multi-file operations as speculated. A similar explanation that most of the overhead associated with TIFPS occurs from setup of the function calls to the security hook implementation on file operations is speculated. Allocating and initializing security data structures does not seem to contribute to the overhead of TIFPS as much as the setup for the security hook function calls.

The detailed test results are captured in Appendix D, Section F.

C. CONCURRENCY TESTS

The objective of the set of concurrency tests is to provide a gauge for the robustness of the TIFPS LSM in handling situations where multiple users with different time attributes request access to the same files and directories. To test concurrent access to files and directories, three user accounts (Sam, Jody, and Don) were created on the

system, each was assigned different time attributes by modifying the time attributes of their respective **.bash_profile** file in their home directories. The test plan and results follow.

1. Concurrency Test Plan

It is expected that in multi-user environments, the system should continue to enforce the time-bases policies to revoke access from users at the appropriate time as well as to properly preserve the time attributes of files copied by each user. The concurrency test plan consists of the following tests scenarios for three test users and is summarized in Table 4-13.

Concurrent read access to a file

- Three users, Sam, Jody, and Don each log into their respective accounts, where each account was modified to have different time attributes by the administrator. Each user then attempts to continuously read the same text file using the command **cat**. When read access is revoked, the revocation time is recorded for each user and compared with the expected revocation time.

Concurrent write access to a file

- Sam, Jody, and Don each log into their respective accounts, where each account has different time attributes preset by the administrator. Each user then attempts to continuously write to the same text file, which is located in a shared directory, by using the command:
- *$ echo "user specific message" >>shared-file.txt*
- When write access is revoked, the revocation time is recorded for each user and compared with the expected revocation time.

Concurrent copy operation of a file

- The three users log into their account, each account preset by the administrator with different time attributes. Each user then attempts to continuously and concurrently read a shared file in order to make a copy of the shared file into their respective home directories. After a period of predefined concurrent access, for example, the time it takes to make 1000 copies of the same file, the time attributes of the copied files for each user is checked and compared with the expected time attributes.

Concurrent write to a shared directory

- The three users log into their respective accounts each of which is preset with different time attributes. Each user then attempts to continuously and concurrently copy their private files into a shared directory. After a period of predefined concurrent writes into the directory, i.e. the time it takes to

copy their private file 1000 times into the shared directory, the time attributes of the copies make by each individual user as well as the shared directory are recorded and compared with expected results.

Table 4-13. Summary of test scripts for concurrency testing

Test ID	Description of concurrency test scenario
G1	Concurrent read of a single file by 3 users with different time attributes
G2	Concurrent write to a single file by 3 users with different time attributes
G3	Concurrent copy of a single file by 3 users with different time attributes
G4	Concurrent write to a shared directory by 3 users with different time attributes

2. Results and Analysis

In the concurrent read access scenario, TIFPS continued to enforce the policy correctly and revokes read access at the proper times from the users when their respective time attributes expired. In the concurrent write access scenario, the file correctly inherited the TIFPS permissions of the user whose time attributes are the most restrictive. At file expiration, the write access was properly revoked for all users. In the concurrent copy scenario, each of the three users' copies of the file in their respective home directories inherited the proper time attributes, i.e. those associated with the individual user. Finally, in the concurrent write to a shared directory scenario, each user's respective file time attributes were preserved as expected. The shared directory also kept its time attributes as expected. See Appendix D, Section H and Section I for test scripts and resulting screenshots of these tests.

D. SUMMARY

In this chapter, test plans and test results for the TIFPS LSM were presented. Access control, performance, and concurrency tests were all part of the test plan. For the most part, the system performed as expected. Problems encountered while performing the access control tests were analyzed and the behavior explained. In some cases, solutions were found and implemented for the problems encountered. For the remaining problems, potential solutions were discussed and are also suggested for future work. It is important to note that the problems discussed were related to the TIFPS implementation as opposed to artifacts of testing. Problems related to testing were resolved in the iterative phases of the testing process.

V. CONCLUSIONS

A. SUMMARY

Based on the TIAC model, TIFPS is a kernel implementation of time-based access control for files and directories in the popular open source Linux operating system. The implementation of access authorization and access control described by TIAC was achieved by utilizing the Linux Security Module framework and implementing the existing security hooks that already reside in the LSM. However, for practical reasons, the system also needs to enforce proper inheritance of time attributes by subjects and objects for copy operations. This requirement presents the challenge of balancing correct security behavior and ensuring availability of system services.

To enforce proper inheritance in such a system, a policy similar to the High Watermark [9] must be implemented. The High Watermark policy can be generally characterized as a policy where a subject's level of access becomes increasingly restrictive as the subject accesses the objects in the system. However, with such a policy, the potential for the system to become so restrictive that the user can not accomplish intended tasks is likely. For example, as a user reads more and more files in the system, his ability to access other files and directories to do useful work in the current session decreases as his time attributes becomes increasingly restrictive.

In Linux, the fork-and-exec paradigm shows potential for solving this dilemma as is evident in the implementation. By forking the parent login shell to a child process and performing read and write operations using the child process, the parent process's time attribute does not become increasingly restrictive. However, the fork-and-exec paradigm introduces additional issues. For example, inheritance of time attributes was not properly enforced in the copy operation performed using pipes to communicate information between sibling processes within the system. Therefore, it is recommended that, for future implementations, the fork-and-exec functionality be examined more closely to ensure that object and subject time attributes are preserved.

B. FUTURE WORK

The TIFPS prototype shows that implementing an access control system based on the TIAC model is feasible for files and directories in Linux. However, by doing so, additional research questions are raised. The following discusses immediate future work to continue the development of and to address the issues related to the prototype. Longer term research related to the topic of time-based access control is also suggested.

1. Prototype Related Work

- As mentioned previously, the fork-and-exec functionality in Linux should be looked at more closely to ensure proper enforcement of the policy when attributes are inherited by new processes. Related to this topic is the **bash** auto-completion for executables problem mentioned in Chapter IV.
- The Unix time is represented by a 32-bit signed integer which allows time specification until 2038. It is expected that Unix-based operating systems will switch to a 64-bit integer for time representation. TIFPS should be modified to support such a change in Unix time.
- TIFPS currently supports only "ext3" file systems. It should be easily modifiable to support other file systems so long as the file system supports extended attributes.
- The **modtime** tool currently does not support recursion into directories for modification of or displaying the time attributes. Adding such support will make the tool more useful for modification of time attributes for entire file trees.

2. Long Term Time-Based Access Control Research Questions

- TIFPS was prototyped to enforce access control locally within a host Linux system. How would such a system be implemented in a networked environment?
- The revocation of access during file modification has the potential to corrupt files as demonstrated in Chapter VI. What is involved in building kernel level support for transactional write operation in Linux?
- What APIs are needed to help applications deal with time based revocation for better usability?
- The TIAC model [1,3] does not consider creation and modification of time attributes. Such actions are necessary in copy operations. For example, in this implementation, a subject copying a source object to a destination object transfers the time attributes from the source to the destination. How can the TIAC model be extended to describe this inheritance policy and to formally check it for consistency?

C. CONCLUSIONS

Temporal access control provides another vector for the management of information. There are many potential applications of such an access control mechanism in civilian and government environments. This simple TIFPS prototype implementation in Linux provides a potential framework for how future time-based access control systems could be built.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. SOURCE CODE

This appendix contains source code for the TIFPS LSM as well as the **modtime** tool.

A. TIFPS LSM SOURCE CODE

linux/security/tifps/tifps_hooks.c

```
/*
 *      Time Interal File Protection System (TIFPS)
 *      Linux Security Module (LSM)
 *
 *      This file contains the TIFPS security hooks function implementations
 *       as well as helper functions used by the LSM to enforce a time based
 *       access control policy on regular files and directories.
 *
 *      It currently only supports ext3 file systems.
 *
 *      Author:  Ken Chiang <kchiang@nps.edu>
 *               Naval Postgraduate School
 *
 *      Last Update:  9/6/06
 *
 *     This program is free software; you can redistribute it and/or modify
 *     it under the terms of the GNU General Public License as published by
 *     the Free Software Foundation; either version 2 of the License, or
 *     (at your option) any later version.
 */

#include <linux/config.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/security.h>
#include <linux/file.h>
#include <linux/fs.h>
#include <linux/mm.h>
#include <linux/mman.h>
#include <linux/mount.h>
#include <linux/xattr.h>
#include <linux/types.h>

#include “tifps_sec_objects.h”

#define XATTR_TIFPS_SUFFIX “tifps”
#define XATTR_NAME_TIFPS XATTR_SECURITY_PREFIX XATTR_TIFPS_SUFFIX
#define TIFPS_XATTR_LEN 23    //tifps format = “:0x00000001:x7FFFFFFF\0”
#define TIFPS_MAX 0x7fffffff
#define TIFPS_MIN 0x00000000


/* ------------------------TIFPS helper functions-------------------------*/

/* tifps_time_to_xattr_value:  converts a set of tifps start and end time
attributes into tifps format string specified by the char * pointer “value”.
Returns 0 on success and appropriate error otherwise.  */
static int tifps_time_to_xattr_value(void **value, uint32_t value_len,
                                     time_t start, time_t end)
{
        char *tmp_string;
        int num_char = 0;
        int rc = 0;

        tmp_string = kmalloc(value_len, GFP_KERNEL);
```

```
        if (!tmp_string  ) {
                rc = -ENOMEM;
                goto out_no_free;
        }

        if (start < TIFPS_MIN || end > TIFPS_MAX){
                rc = -EINVAL;
                 goto out;
        }

        /* change the format string to :0x%016x:0x%016x” for 8-byte
         * time support in the future*/
        num_char = snprintf(tmp_string  , value_len,
                     “:0x%08x:0x%08x”, start, end);

        if (num_char != value_len-1){
                rc = -EINVAL;
                goto out;
        }

        tmp_string[value_len-1] = 0;
        memcpy(*value, tmp_string, value_len);

out:
        kfree(tmp_string);
out_no_free:
        return rc;
}

/* tifps_get_times:  Given a tifps formated string “value”, parse the string
 * for the tifps start and end times.
 * Returns 0 on success and appropriate error otherwise. */
static int tifps_get_times(char *value, uint32_t value_len,
                           time_t *start, time_t *end)
{
        char *tifps_string;
        time_t tifps_start;
        time_t tifps_end;
        char *tifps_string_ptr, *p, *d;
        int rc = -EINVAL;

        /* copy the string so that we can modify the copy as we parse it.
           The string should already be null terminated, but we append a
           null suffix to the copy to avoid problems with the existing
           attr package, which does not view the null terminator as part
           of the attribute value. */
        tifps_string = kmalloc(value_len, GFP_KERNEL);
        if (!tifps_string) {
                rc = -ENOMEM;
                goto out_no_free;
        }
        memcpy(tifps_string, value, value_len);
        tifps_string[value_len] = 0;

        tifps_string_ptr = (char *) tifps_string+1; /*skip the first “:”*/

        p = tifps_string_ptr;
        while (*p && *p != ':')
                p++;

        if (*p == 0)
                goto out;
        *p++ = 0;

        tifps_start = simple_strtoul(tifps_string_ptr, &d, 0);

        if (tifps_start < TIFPS_MIN)
                goto out;

        *start = tifps_start;
```

```
        tifps_string_ptr = p;
        while (*p)
                p++;
        *p++ = 0;

        tifps_end = simple_strtoul(tifps_string_ptr, &d, 0);

        if (tifps_end > TIFPS_MAX)
                goto out;

        *end = tifps_end;
        rc = 0;
out:
       kfree(tifps_string);

out_no_free:
        return rc;
}

/* tifps_helper_task_alloc_security; the tifps_task_alloc_security hook
 * calls this function.  It is defined as a helper function because
 * inode_permission also calls it if a task does not have a security struct
 * associated with it.
 * Note, this security hook is normally called during the copy_process()
 * function, where the process has not been started.  Therefore, we
 * will inherit from the “current” task rather than the “parent”
 * task.  In the case it is called from inode_permission, the security
 * should be null and the max range for tifps attributes set for the task.
 * Returns 0 if success and appropriate error otherwise.
 */
static int tifps_helper_task_alloc_security (struct task_struct *task)
{
       struct tifps_task_security_struct *tsec;
       struct tifps_task_security_struct *parent_tsec;

       tsec = kzalloc(sizeof(struct tifps_task_security_struct), GFP_KERNEL);
       if (!tsec)
              return -ENOMEM;

       init_MUTEX(&tsec->sem);
       tsec->task = task;

       /* inherit time attributes from parent task, i.e. the current process
        * that we are copying */
       parent_tsec = current->security;
       if ( parent_tsec ){
              tsec->tifps_start = parent_tsec->tifps_start;
              tsec->tifps_end = parent_tsec->tifps_end;

       }
       else{
              tsec->tifps_start = TIFPS_MIN;
              tsec->tifps_end = TIFPS_MAX;
       }

       task->security = tsec;

       return 0;
}

/* tifps_update_task_security:
 * To prevent information from being copied to pass the TIFPS system,
 * anytime a task reads a file, its tifps attributes must be updated
 * to reflect the more restricted time interval. */
static void tifps_update_task_security(struct tifps_inode_security_struct *isec)
{
       struct tifps_task_security_struct *tsec = current->security;
       time_t old_start = tsec->tifps_start;
       time_t old_end = tsec->tifps_end;
       time_t new_start = isec->tifps_start;
       time_t new_end = isec->tifps_end;
```

```
        if (new_start < TIFPS_MIN || new_end > TIFPS_MAX)
                return;
        else{
                if (new_start > old_start)
                        tsec->tifps_start = new_start;
                if (new_end < old_end)
                        tsec->tifps_end = new_end;
        }

        return;
}

/* tifps_update_inode_security:
 * This method updates the TIFPS attributes for an inode.
 * It is used to enforce proper inheritance of time attributes
 * of files in copy operations.
 */
static void tifps_update_inode_security(
        struct tifps_inode_security_struct *isec, struct dentry *dentry )
{
        struct tifps_task_security_struct *tsec = current->security;
        time_t old_start = isec->tifps_start;
        time_t old_end = isec->tifps_end;
        time_t new_start = tsec->tifps_start;
        time_t new_end = tsec->tifps_end;
        struct inode_operations *i_ops = isec->inode->i_op;
        char * tifps_string;
        umode_t mode = isec->inode->i_mode;

        if (new_start < TIFPS_MIN || new_end > TIFPS_MAX)
                return;
        else{

                if (new_start > old_start)
                        isec->tifps_start = new_start;
                if (new_end < old_end)
                        isec->tifps_end = new_end;

                /* if this inode describes a fifo pipe, do not set
                 *  extended attributes, because pipe file systems do
                 *  not support extended attributes */
                if ( S_ISFIFO(mode) )
                        goto out;

                tifps_string=kmalloc(TIFPS_XATTR_LEN, GFP_KERNEL);
                tifps_time_to_xattr_value(&tifps_string, TIFPS_XATTR_LEN,
                                          isec->tifps_start, isec->tifps_end);
                i_ops->setxattr(dentry, XATTR_NAME_TIFPS,
                                tifps_string, TIFPS_XATTR_LEN, 0);
                kfree(tifps_string);
        }
out:
        return;
}

/* tifps_enforcer:
 * The main access control policy enforcer return 0 if allowed, -EPERM
 * otherwise.  Update tasks for every read operation to take on more
 * restrictive TIFPS attributes and Updates inodes for every write
 * operation to files.
*/
static int tifps_enforcer (struct tifps_task_security_struct *tsec,
                           struct tifps_inode_security_struct *isec,
                           int mask, struct dentry *dentry)
{
        struct timeval current_time;
        umode_t mode;
        int rc = 0;

        /* if root user with CAP_SYS_ADMIN capability, allow */
```

```
        if (capable(CAP_SYS_ADMIN)){
               rc = 0;
               goto out;
        }

        /* update the task attributes if the access mode requested
         * is read and the object is a regular file or fifo pipe.
         * Linux fork and exec paradigm prevents a task from becoming
         * overly restrictive as it read more files, avoiding a
         * denial of service condition where a user's login shell
         * becomes increasingly restrictive.  */
        mode = isec->inode->i_mode;

        if ( (S_ISREG(mode)||S_ISFIFO(mode))  && mask & MAY_READ){
               down_interruptible(&tsec->sem);
               tifps_update_task_security(isec);
               up(&tsec->sem);
        }

        do_gettimeofday(&current_time);

        if (current_time.tv_sec >= tsec->tifps_start &&
            current_time.tv_sec < tsec->tifps_end)
                rc = 0;
        else{
                rc = -EPERM;
               goto out;
        }

        if (current_time.tv_sec >= isec->tifps_start &&
            current_time.tv_sec < isec->tifps_end)
                rc = 0;
        else{
               rc = -EPERM;
               goto out;
        }

        /* Update time attribute only if the file is a regular file or
         * fifo pipe (not directories), and the task is writing or
         * appending to the object.  */
        if ( (S_ISREG(mode)||S_ISFIFO(mode)) &&
             (mask & MAY_WRITE || mask & MAY_APPEND) ){
               down_interruptible(&isec->sem);
               tifps_update_inode_security( isec, dentry);
               up(&isec->sem);
        }

out:
        return rc;
}

/* tifps_inode_has_perm:
 * Checks with the enforcer whether access to an inode is allowed.
 * This function is called by the security hook tifps_inode_permission
 * during initial opening of files.  It is also called by tifps_file_has_perm
 * for ongoing file descriptor access. */
static int tifps_inode_has_perm( struct inode *inode, int mask )
{
        struct tifps_task_security_struct *tsec = current->security;
        struct tifps_inode_security_struct *isec = inode->i_security;
        umode_t mode = inode->i_mode;
        struct dentry *dentry;
        struct list_head *head;
        struct inode_operations *i_ops;
        char *tifps_string;
        time_t new_start =TIFPS_MIN;
        time_t new_end =TIFPS_MAX;
        int rc = 0;

        /* We are only interested in controlling read, write, and execute of
         * regular files and directories for this prototype.
```

```
         * We also include fifo pipes as they can be used to copy
         * the contents of files. */
        if (!S_ISREG(mode) && !S_ISDIR(mode) && !S_ISFIFO(mode) )
                goto out_no_free;

        if (!tsec){
                tifps_helper_task_alloc_security(current);
        }
        /* Is this needed? */
        tsec = current->security;

        head = inode->i_dentry.next;
        dentry = list_entry(head, struct dentry, d_alias);

        /* If the operation is a pipe operation, no need to get
         * extended attributes, just call tifps_enforcer
         * to properly update the task and inode security
         * data structures.  */
        if (S_ISFIFO(mode)){
                rc = tifps_enforcer ( tsec, isec, mask, dentry );
                goto out_no_free;
        }

        /* Our prototype only controls ext3 file systems at the moment,
         * but it can easily support any file systems that support
         * extended attributes. */
        if ( strncmp(dentry->d_sb->s_type->name, “ext3”, 4 ) ) {
                goto out_no_free;
        }

        tifps_string = kzalloc(TIFPS_XATTR_LEN, GFP_KERNEL);
        if (!tifps_string) {
                return -ENOMEM;
        }

        i_ops = inode->i_op;
        i_ops->getxattr(dentry, XATTR_NAME_TIFPS,
                            tifps_string, TIFPS_XATTR_LEN);

        if (tifps_string[0] !=':'){
                rc = 0;
                goto out;
        }

        rc = tifps_get_times(tifps_string, TIFPS_XATTR_LEN,
                            &new_start, &new_end);
        if(rc){
                printk(“get_times error\n”);
                rc = -EINVAL;
                goto out;
        }
        isec->tifps_start = new_start;
        isec->tifps_end = new_end;

        rc = tifps_enforcer( tsec, isec, mask, dentry );

out:
        kfree(tifps_string);
out_no_free:
         return rc;
}


/* tifps_file_has_perm:
 * Checks with the enforce whether ongoing access to a file is permitted.
 *  Calls tifps_inode_has_perm.  */
static int tifps_file_has_perm(       struct file *file, int mask)
{
        struct dentry * dentry = file->f_dentry;
        struct inode * inode = dentry->d_inode;
         int rc = 0;
```

```
        rc = tifps_inode_has_perm( inode, mask );

        return rc;
}
/* ------------ End of TIFPS helper functions-----------------------------*/



/* ------------------------------TIFPS security hooks-----------------------
 * The following security hook functions are provided for TIFPS access control.
 * These defined functions plus others  are called by the kernel at strategic
 * locations throughout the kernel as part of the the Linux Security Module.
 * See include/linux/security.h for a list and description of the Linux
 * Security Module hooks.  If a security hook function is not defined
 * specifically, the result is a usually a nop defined in
 * linux/security/dummy.c
*/

int tifps_inode_alloc_security(struct inode *inode)
{
        struct tifps_inode_security_struct *isec;

        isec = kzalloc(sizeof(struct tifps_inode_security_struct), GFP_KERNEL);
        if (!isec)
                return -ENOMEM;

        isec->inode = inode;
        init_MUTEX(&isec->sem);
        inode->i_security = isec;

        isec->tifps_start = TIFPS_MIN;
        isec->tifps_end = TIFPS_MAX;

        return 0;
}
EXPORT_SYMBOL(tifps_inode_alloc_security);

void tifps_inode_free_security(struct inode *inode)
{

        struct tifps_inode_security_struct *isec = inode->i_security;

        inode->i_security = NULL;
        kfree (isec);
}
EXPORT_SYMBOL(tifps_inode_free_security);

int tifps_inode_init_security(struct inode *inode, struct inode *dir,
                                    char **name, void **value, size_t *len)
{
        struct tifps_task_security_struct *tsec;
        struct tifps_inode_security_struct *isec;
        char * tifps_string;

        /* default allow access by setting access time to min and max values */
        time_t new_start = TIFPS_MIN;
        time_t new_end = TIFPS_MAX;

        int rc = 0;
        char *namep = NULL;
        char *valuep;

        tifps_string = kzalloc(TIFPS_XATTR_LEN, GFP_KERNEL);
        if (!tifps_string){
                return -ENOMEM;
        }
        tsec = current->security;

        isec = inode->i_security;
```

```
        if (!isec){
                rc= tifps_inode_alloc_security(inode);
                isec = inode->i_security;
        }
        if (rc){
                return rc;
        }


        new_start = (time_t)tsec->tifps_start;
        new_end = (time_t)tsec->tifps_end;

        isec->tifps_start = new_start;
        isec->tifps_end = new_end;

        if (name) {
                namep = kstrdup(XATTR_TIFPS_SUFFIX, GFP_KERNEL);
                if (!namep){
                        rc = -ENOMEM;
                        goto out;
                }
                *name = namep;
        }

        if (value) {
                valuep = kmalloc(TIFPS_XATTR_LEN, GFP_KERNEL);
                if (!valuep){
                        rc = -ENOMEM;
                        kfree(namep);
                        goto out;
                }
                rc = tifps_time_to_xattr_value(&valuep, TIFPS_XATTR_LEN,
                                        new_start, new_end);
                if(rc){
                        kfree(namep);
                        kfree(valuep);
                        goto out;
                }
                *value = valuep;
        }
        if (len) {
                *len = TIFPS_XATTR_LEN;
        }
out:
        kfree(tifps_string);
        return rc;
}
EXPORT_SYMBOL(tifps_inode_init_security);

int tifps_inode_permission (struct inode *inode, int mask,
                                    struct nameidata *nd)
{
        struct tifps_inode_security_struct *isec = inode->i_security;
        int rc = 0;

        if (!mask) {
                /* No permission to check.  Access allowed */
                return 0;
        }

        if (!isec) {
                rc = tifps_inode_alloc_security(inode);
                isec = inode->i_security;
        }

        if (rc){
                return rc;
        }

        rc = tifps_inode_has_perm( inode, mask );
```

```
        return rc;
}
EXPORT_SYMBOL(tifps_inode_permission);

void tifps_inode_post_setxattr(struct dentry *dentry, char *name,
                                    void *value, size_t size, int flags)
{
        struct inode *inode = dentry->d_inode;
        struct tifps_inode_security_struct *isec;
        time_t new_start;
        time_t new_end;
        int rc;

        if (strcmp(name, XATTR_NAME_TIFPS)) {
                /* Not a TIFPS attribute, do nothing. */
                return;
        }

        rc = tifps_get_times((char *)value, size, &new_start, &new_end);

        if (rc) {
                printk(KERN_WARNING "%s: error getting TIFPS attributes "
                        "%s, rc= %d\n", __FUNCTION__, (char*)value, -rc);
                return;
        }

        isec = inode->i_security;
        if (!isec){
                rc = tifps_inode_alloc_security(inode);
                isec = inode->i_security;
        }
        if (rc){
                printk(KERN_WARNING "%s: error allocating security struct"
                        "%s, rc= %d\n", __FUNCTION__, (char*)value, -rc);
                return;
        }
        isec->tifps_start = new_start;
        isec->tifps_end = new_end;
        return;
}
EXPORT_SYMBOL(tifps_inode_post_setxattr);


/* inode_setsecurity is very similar to inode_post_setxattr, it is called
by vfs_setxattr in the event that the setxattr function is not define for
an inode in a particular file system. */
int tifps_inode_setsecurity (struct inode *inode, const char *name,
                                  const void *value, size_t size, int flags)
{
        struct tifps_inode_security_struct *isec;
        time_t new_start;
        time_t new_end;
        int rc = 0;

        if (strcmp(name, XATTR_NAME_TIFPS)) {
                /* Not a TIFPS attribute, do nothing. */
                return rc;
        }

        rc = tifps_get_times((char *) value, size, &new_start, &new_end);
        if (rc) {
                printk(KERN_WARNING "%s: error getting TIFPS attributes "
                        "%s, rc= %d\n", __FUNCTION__, (char*)value, -rc);
                return -EINVAL;
        }

        isec = inode->i_security;
        if (!isec){
                rc= tifps_inode_alloc_security(inode);
                isec = inode->i_security;
        }
```

```
        if (rc){
                return rc;
        }

        isec->tifps_start = new_start;
        isec->tifps_end = new_end;
        return rc;
}

int tifps_file_permission(struct file *file, int mask)
{
        struct inode *inode = file->f_dentry->d_inode;
        int rc = 0;

        if (!mask) {
                /* No permission to check. permission allowed */
                return 0;
        }

        if (!inode->i_security) {
                rc = tifps_inode_alloc_security(inode);
        }

        if (rc){
                return rc;
        }

        return tifps_file_has_perm( file, mask);
}
EXPORT_SYMBOL(tifps_file_permission);


int tifps_task_alloc_security (struct task_struct *task)
{
        return tifps_helper_task_alloc_security(task);
}
EXPORT_SYMBOL(tifps_task_alloc_security);

void tifps_task_free_security (struct task_struct *task)
{
        struct tifps_task_security_struct *tsec = task->security;

        task->security = NULL;
        kfree (tsec);
}
EXPORT_SYMBOL(tifps_task_free_security);

static struct security_operations tifps_security_ops = {
        .inode_alloc_security =              tifps_inode_alloc_security,
        .inode_free_security =               tifps_inode_free_security,
        .inode_init_security =       tifps_inode_init_security,
        .inode_permission =          tifps_inode_permission,
        .inode_post_setxattr =               tifps_inode_post_setxattr,
         .inode_setsecurity =        tifps_inode_setsecurity,

        .file_permission =           tifps_file_permission,

        .task_alloc_security =       tifps_task_alloc_security,
        .task_free_security =        tifps_task_free_security,
};

/* flag to keep track of how the tifps security module was registered */
static int secondary;

static int __init tifps_init (void)
{
        printk(KERN_ALERT"\nInitializing TIFPS Linux Security Module"
                " - created by Ken Chiang- Naval Postgraduate School\n");

        /* allocate a security struct for the initial task */
        if (tifps_task_alloc_security(current))
```

```
                panic("TIFPS: Failed to initialize the initial task.\n");

        /* register tifps with the security framework */
        if (register_security (&tifps_security_ops)) {
                /* try to register with primary module */
                if (mod_reg_security(KBUILD_MODNAME, &tifps_security_ops)) {
                        printk (KERN_INFO "Registration of TIFPS with primary "
                                " security module failed\n");
                        return -EINVAL;
                }
                secondary = 1;
        }

        printk (KERN_ALERT "\n...TIFPS LSM Initialized %s %s\n",
                secondary ? " as secondary" : "as primary",
                "security module.");
        return 0;
}

static void __exit tifps_exit (void)
{
        /* unregister and cleanup at module exit */
        printk(KERN_ALERT "\n*** TIFPS LSM removed ***\n");
        if (secondary) {
                /* print kernel error message if unregistering from primary
                   module fails */
                if (mod_unreg_security (KBUILD_MODNAME, &tifps_security_ops))
                        printk (KERN_INFO "Failure unregistering TIFPS "
                                "with primary module.\n");
        }

        if (unregister_security (&tifps_security_ops)) {
                printk (KERN_INFO
                        "Failure unregistering Time Interval File"
                        " Protection System with the kernel\n");
        }
}

security_initcall (tifps_init);
module_exit (tifps_exit);

MODULE_DESCRIPTION("Experiemental Time Interval File Protection System LSM");
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Ken Chiang - Naval Postgraduate School");
```

linux/security/tifps/include/tifps_sec_objects.h

```
/*  Time Interval File Protection System (TIFPS) security module
 *
 *  This file contains definitions for the TIFPS security data structures for
 *   kernel objects.
 *
 *  Author:  Ken Chiang, <kchiang@nps.edu>
 *
 *  Last update:  9/6/06
 *
 *     This program is free software; you can redistribute it and/or modify
 *     it under the terms of the GNU General Public License version 2,
 *     as published by the Free Software Foundation.
 */

#ifndef _TIFPS_SEC_OBJECTS_H_
#define _TIFPS_SEC_OBJECTS_H_

#include <linux/list.h>
#include <linux/fs.h>
#include <linux/sched.h>
#include <linux/in.h>
#include <linux/types.h>
```

```
#include <asm/semaphore.h>

struct tifps_task_security_struct {
        struct task_struct *task;     /* back pointer to task object */
        struct semaphore sem;
        time_t tifps_start;
        time_t tifps_end;
};

struct tifps_inode_security_struct {
        struct inode *inode;                /* back pointer to inode object */
        struct semaphore sem;
        time_t tifps_start;
        time_t tifps_end;
};
#endif /* _TIFPS_SEC_OBJECTS_H_*/
```

linux/security/Kconfig

```
#
# Security configuration
#

menu “Security options”

config KEYS
        bool “Enable access key retention support”
        help
          This option provides support for retaining authentication tokens and
          access keys in the kernel.

          It also includes provision of methods by which such keys might be
          associated with a process so that network file systems, encryption
          support and the like can find them.

          Furthermore, a special type of key is available that acts as keyring:
          a searchable sequence of keys. Each process is equipped with access
          to five standard keyrings: UID-specific, GID-specific, session,
          process and thread.

          If you are unsure as to whether this is required, answer N.

config KEYS_DEBUG_PROC_KEYS
        bool “Enable the /proc/keys file by which all keys may be viewed”
        depends on KEYS
        help
          This option turns on support for the /proc/keys file through which
          all the keys on the system can be listed.

          This option is a slight security risk in that it makes it possible
          for anyone to see all the keys on the system. Normally the manager
          pretends keys that are inaccessible to a process don't exist as far
          as that process is concerned.

config SECURITY
        bool “Enable different security models”
        depends on SYSFS
        help
          This allows you to choose different security modules to be
          configured into your kernel.

          If this option is not selected, the default Linux security
          model will be used.

          If you are unsure how to answer this question, answer N.

config SECURITY_NETWORK
        bool “Socket and Networking Security Hooks”
        depends on SECURITY
```

```
        help
          This enables the socket and networking security hooks.
          If enabled, a security module can use these hooks to
          implement socket and networking access controls.
          If you are unsure how to answer this question, answer N.

config SECURITY_NETWORK_XFRM
        bool "XFRM (IPSec) Networking Security Hooks"
        depends on XFRM && SECURITY_NETWORK
        help
          This enables the XFRM (IPSec) networking security hooks.
          If enabled, a security module can use these hooks to
          implement per-packet access controls based on labels
          derived from IPSec policy.  Non-IPSec communications are
          designated as unlabelled, and only sockets authorized
          to communicate unlabelled data can send without using
          IPSec.
          If you are unsure how to answer this question, answer N.

config SECURITY_CAPABILITIES
        tristate "Default Linux Capabilities"
        depends on SECURITY
        help
          This enables the "default" Linux capabilities functionality.
          If you are unsure how to answer this question, answer Y.

config SECURITY_ROOTPLUG
        tristate "Root Plug Support"
        depends on USB && SECURITY
        help
          This is a sample LSM module that should only be used as such.
          It prevents any programs running with egid == 0 if a specific
          USB device is not present in the system.

          See <http://www.linuxjournal.com/article.php?sid=6279> for
          more information about this module.

          If you are unsure how to answer this question, answer N.

config SECURITY_SECLVL
        tristate "BSD Secure Levels"
        depends on SECURITY
        select CRYPTO
        select CRYPTO_SHA1
        help
          Implements BSD Secure Levels as an LSM.  See
          <file:Documentation/seclvl.txt> for instructions on how to use this
          module.

          If you are unsure how to answer this question, answer N.

source security/selinux/Kconfig

source security/tifps/Kconfig

endmenu
```

linux/security/Makefile

```
#
# Makefile for the kernel security code
#

obj-$(CONFIG_KEYS)                      += keys/
subdir-$(CONFIG_SECURITY_SELINUX)       += selinux

# if we don't select a security model, use the default capabilities
ifneq ($(CONFIG_SECURITY),y)
obj-y          += commoncap.o
```

```
endif

# Object file lists
obj-$(CONFIG_SECURITY)                += security.o dummy.o inode.o
# Must precede capability.o in order to stack properly.
obj-$(CONFIG_SECURITY_SELINUX)                += selinux/built-in.o

#Chiang-NPS-TIFPS
obj-$(CONFIG_SECURITY_TIFPS)          += tifps/

obj-$(CONFIG_SECURITY_CAPABILITIES)   += commoncap.o capability.o
obj-$(CONFIG_SECURITY_ROOTPLUG)               += commoncap.o root_plug.o
obj-$(CONFIG_SECURITY_SECLVL)         += seclvl.o
```

linux/security/tifps/Kconfig

```
config SECURITY_TIFPS
        tristate "NPS TIFPS (Experimental)"
        depends on SECURITY && EXPERIMENTAL &&!SECURITY_SELINUX &&!SECURITY_CAPABILITIES \
        && !SECURITY_ROOTPLUG && !SECURITY_SECLVL
        default n
        help
          This selects the experimental Linux Security Module for time-based
          access control to files.
          Developed as a thesis project at the Naval Postgraduate School.
          WARNING: This security module is highly experimental, only ext3
          file systems are currently supported.  File corruption may
          occur when a file expires during a write operation.

          This security module does not work with other security modules,
          do not build into the kernel other security modules if you want
          to test TIFPS.

          If you are unsure how to answer this question, answer N.
```

linux/security/tifps/Makefile

```
# Chiang-NPS-TIFPS
# Makefile for building the TIFPS module as part of the kernel tree.
#

obj-$(CONFIG_SECURITY_TIFPS) := tifps.o

tifps-objs := tifps_hooks.o

EXTRA_CFLAGS += -Isecurity/tifps/include
```

B. MODTIME TOOL SOURCE CODE

modtime

```
#!/bin/bash
# Chiang-NPS
#      Time Interval File Protection System (TIFPS)
#
#      This bash script is a front end to manipulating the time attributes
#       associated with files and directory for access control with TIFPS.
#
#       Note:  The script requires the extended attributes tools
#                 "getfattr" and "setfattr"
#                  and perl to run.
#
#      Author:  Ken Chiang <kchiang@nps.edu>
```

```
#               Naval Postgraduate School
#
#       Last update:  9/6/06
#

MINTIME=0
MAXTIME=2147483647

ERROR=7
SU_ERROR=8
DISPLAYFLAG=0
DELETEFLAG=0
ABSOLUTE_START=0
ABSOLUTE_END=0
STARTMODIFIED=0
ENDMODIFIED=0
STARTMODS=0
ENDMODS=0
NOW=`date -d now +%s`

# Function that tells users how to use the program when incorrectly used.
function error
{
    echo “Usage:”
    echo “  Setting time attributes:”
    echo “    By absolute time:”
    echo “      modtime -a<start date-string> -A<end date-string> <file|directory>“
    echo “      Example:  modtime now '09/22/2006 12:00:00' TIMED-file.txt”
    echo “    By relative time:”
    echo “      modtime <relative time flags> <file|directory>“
    echo “       where relative time flags are summarized below.”
    echo “         -w<weeks from now to allow>, -W<weeks from now to revoke>“
    echo “         -d<days from now to allow>, -D<days from now to revoke>“
    echo “         -h<hours from now to allow>, -H<hours from now to revoke>“
    echo “         -m<minutes from now to allow>, -M<minutes from now to revoke>“
    echo “         -s<seconds from now to allow>, -S<seconds from now to revoke>“
    echo “        Note: negative integers represent an earlier time from now.”
    echo “”
    echo “  Deleteing time attributes:”
    echo “      modtime -x <file|directory>“
    echo “”
    echo “  Getting time attributes:”
    echo “      modtime -g <file|directory>“
    echo “”
    echo “”
    echo “  Note:  To change/set/delete the time security attributes, you must be “
    echo “         root or a user with CAP_SYS_ADMIN capability, see man (5)”
    echo “         attr for more information in extended attributes.”
    echo “”
    exit $ERROR
}

function super_user_error
{
    echo “You must be root or a super user with CAP_SYS_ADMIN capability”
    echo “ to set/modify/delete the time attributes of the target”
    echo “”
    exit $SU_ERROR
}


function display_result  #Expects 1 argument; the target file name
{
    TARGET=$1

    if [ $NEW -gt 0 ];then
              return 0;
    fi

    echo “Target:  $TARGET”
    DISPLAYRESULT=`perl -e “print scalar localtime $TIFPS_START”`
```

```
    echo “Grant access on:  $DISPLAYRESULT”
    DISPLAYRESULT=`perl -e “print scalar localtime $TIFPS_END”`
    echo “Revoke access on: $DISPLAYRESULT”
    echo
}

function do_it  #Expects 1 argument; the target file name.
{
    NEW=0
    TARGET=$1

    if  ! [ -e $TARGET ];then
        echo “file or directory: $TARGET does not exist”
        error
    fi

    if [ $DELETEFLAG -gt 0 ]; then
        if [ $EUID -gt 0 ]; then
            super_user_error
        else
            setfattr -x security.tifps $TARGET
            if [ $? -eq 0 ]; then
               echo “$TARGET:  TIFPS attributes deleted.”
            fi
            return;
        fi
    fi

    TIFPS_STRING=`getfattr -n security.tifps $TARGET|grep security.tifps`
    if [ -z $TIFPS_STRING ];then
        echo “$TARGET does not currently have accessible TIFPS attributes”
        NEW=1
    fi

    TIFPS_ATTR=${TIFPS_STRING#*:}

    if [ $ABSOLUTE_START -eq 0 ]; then
        TIFPS_START_HEX=${TIFPS_ATTR%:*}
        TIFPS_START=`printf “%d\n” $TIFPS_START_HEX`
    fi
    if [ $ABSOLUTE_END -eq 0 ]; then
        TIFPS_ENDSTRING=${TIFPS_ATTR#*:}
        TIFPS_END_HEX=${TIFPS_ENDSTRING%\\*}
        TIFPS_END=`printf “%d\n” $TIFPS_END_HEX`
    fi

    if [ $DISPLAYFLAG -gt 0 ]; then
        display_result $TARGET
    else

        if [ $EUID -gt 0 ]; then
            super_user_error
        fi

        if [ $STARTMODIFIED -gt 0 ]; then
            let “TIFPS_START=$STARTMODS+$NOW”
        fi

        if [ $ENDMODIFIED -gt 0 ]; then
            let “TIFPS_END=$ENDMODS+$NOW”
        fi

        if [ $TIFPS_START -gt $MAXTIME ] || [ $TIFPS_START -lt $MINTIME ]; then
            echo “start time out of range”
            error
        fi
        if [ $TIFPS_END -lt $MINTIME ] || [ $TIFPS_END -gt $MAXTIME ];then
            echo “end time out of range”
            error
        fi
        if [ $TIFPS_START -gt $TIFPS_END ]; then
```

```
            echo “Invalid time range”
            error
        fi

        TIFPS_START_HEX=`printf “0x%x\n” $TIFPS_START`
        TIFPS_END_HEX=`printf “0x%x\n” $TIFPS_END`
        `setfattr -n security.tifps -v “:$TIFPS_START_HEX:$TIFPS_END_HEX\000” $TARGET`
        NEW=0
        display_result $TARGET
    fi
}

#check number of arguments
NUMARGS=$#                #get number of arguments
if [ $NUMARGS -lt 2  ];then
    error
fi

# parse the option flags
while getopts “:a:A:w:W:d:D:h:H:m:M:s:S:gx” Option
  do
  case $Option in
      a) TIFPS_START=`date -d “$OPTARG” +%s`
        if [ $? -gt 0 ]; then
            error
        fi
         ABSOLUTE_START=1;;
      g) DISPLAYFLAG=1
         break;;
      x) DELETEFLAG=1
         break;;
      w) let “STARTMODS+=$OPTARG*7*24*60*60”
         STARTMODIFIED=1;;
      d) let “STARTMODS+=$OPTARG*24*60*60”
         STARTMODIFIED=1;;
      h) let “STARTMODS+=$OPTARG*60*60”
         STARTMODIFIED=1;;
      m) let “STARTMODS+=$OPTARG*60”
         STARTMODIFIED=1;;
      s) let “STARTMODS+=$OPTARG”
         STARTMODIFIED=1;;
      A) TIFPS_END=`date -d “$OPTARG” +%s`
         if [ $? -gt 0 ]; then
             error
         fi
         ABSOLUTE_END=1;;
      W) let “ENDMODS+=$OPTARG*7*24*60*60”
         ENDMODIFIED=1;;
      D) let “ENDMODS+=$OPTARG*24*60*60”
         ENDMODIFIED=1;;
      H) let “ENDMODS+=$OPTARG*60*60”
         ENDMODIFIED=1;;
      M) let “ENDMODS+=$OPTARG*60”
         ENDMODIFIED=1;;
      S) let “ENDMODS+=$OPTARG”
         ENDMODIFIED=1;;
      *) echo “Unimplemented option chosen.”;;
  esac
done

# Decrements the argument pointer, so it points to next argument.
shift $(($OPTIND - 1))

# check that target arguments for files/directories are given
if [ “$1” = “” ]; then
    error
fi

#set or get attributes for all arguments
for arg in $*
do
```

```
  do_it $arg
done

exit 0
```

modtime_install.sh

```
#!/bin/bash

# run this install script as root
#   To install modtime:
#      ./modtime_install.sh -i
#
#   To remove modtime:
#      ./modtime_install.sh -u


function check_dependency
{
    echo “Checking for dependencies...”

    which setfattr
    if [ $? -gt 0 ]; then
        echo “You need to install the setfattr and getfattr tools from the attr package
first”
        exit
    fi

    which getfattr
    if [ $? -gt 0 ]; then
        echo “You need to install the setfattr and getfattr tools from the attr package
first”
        exit
    fi
    which perl
    if [ $? -gt 0 ]; then
        echo “You need to install perl first”
        exit
    fi
}


while getopts “:iu” Option
  do
  case $Option in
      u) rm -f /usr/bin/modtime
         rm -f /usr/share/man/man1/modtime.1.gz
         if [ $? -eq 0 ]; then
             echo “Uninstall sucessful!!”
         else
             echo “Uninstall failed! You can manually remove the modtime from the”
             echo “/usr/bin/ directory and the modtime.1.gz file from the “
             echo “/usr/share/man/man1/ directory.”
         fi
         exit;;

      i) check_dependency
         cp -f modtime /usr/bin/
         cp -f modtime.1.gz /usr/share/man/man1/
         if [ $? -eq 0 ]; then
             echo “Install successful!!”
         else
             echo “Uninstall failed!  Copy the modtime to /usr/bin and “
             echo “modtime.1.gz to the /usr/share/man/man1/ directories.”
         fi
         exit;;
      *) echo “This script only takes -u or -i as flags”

  esac
done
```

APPENDIX B. INSTALLATION GUIDE

This is a brief description of how to patch, compile, and install both the TIFPS Linux Security Module and the **modtime** tool. The latter is used to get and set the time attributes for use with the TIFPS LSM.

A. INSTALLING TIFPS MODULE

1. Download and install the Fedora Core 5 (FC5) Linux operating system. The TIFPS Linux Security Module (LSM) should work for any distribution of Linux that supports the LSM framework. Specifically, this implementation was developed with kernel version 2.6.15 using the Fedora Core 5 distribution. It was compiled and tested on an i686 machine.

2. During operating system (OS) installation, make sure to also install the perl and attr packages. These are required for the **modtime** tool to work. Note: In FC5, both should be installed by default.

3. After the OS install, download and install the kernel source code. For FC5:

   a. First, make a note of the kernel version installed by typing:

      $ uname -r

   b. Then, download the kernel source package (kernel-<version>.src.rpm) from:

   http://download.fedora.redhat.com/pub/fedora/linux/core/5/source/SRPMS/

   c. As root, install the source rpm by:

      # rpm -Uvh kernel-<version>.src.rpm

   d. Build the kernel source:

      # cd /usr/src/redhat/SPECS

      # rpmbuild -bp --target $(uname –m) kernel-2.6.spec

e. The source should now be installed in /usr/src/redhat/BUILD/kernel-<version>/linux-2.6.15.<arch>/ directory, create a symlink to this source directory:

 # ln –s /usr/src/redhat/BUILD/kernel-<version>/linux-2.6.15.<arch> \ /usr/src/linux

4. Patch the kernel with TIFPS with the following steps:

   a. As root, change into the kernel source directory:

      # cd /usr/src/linux

   b. Patch the kernel source with the tifps_patch by:

      # cp <path to tifps_patch_kernel-2.6.15_090606 on CD1> /usr/src/linux

      # patch -p1 –i /usr/src/linux/ tifps_patch_kernel-2.6.15_090606

   c. To revert back to original kernel, type:

      # patch -p1 -R -i /usr/src/linux/tifps_patch_kernel-2.6.15_090606

5. Configure the new kernel with tifps selected as a module:

   a. As root, change into kernel source directory and run the following command to keep the existing kernel configuration:

      # make oldconfig

   b. Answer ‘N’ for any new kernel options available.

   c. Next, run:

      # make menuconfig

   d. Select the kernel options required to support the hardware associated with the system upon which it will execute. The default should work.

   e. Go to the “Security options” option using arrow keys and using the space bar to select options (See Figure B-1):

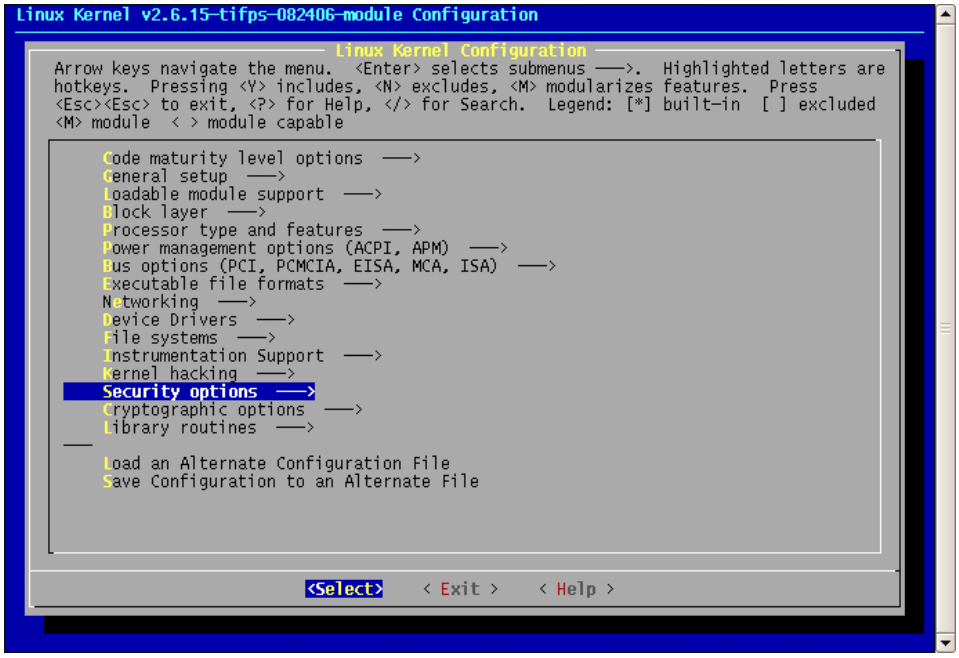


Figure B-1. Select “Security options”

f. Unselect NSA SELinux Support.

g. Unselect all other security models or select them as modules:

   i. Default Linux Capabilities

   ii. BSD Secure Levels

   iii. Root Plug Support (will appear only if USB support is selected)

h. Select “NPS TIFPS (Experimental)” as a module. See Figure B-2 below.

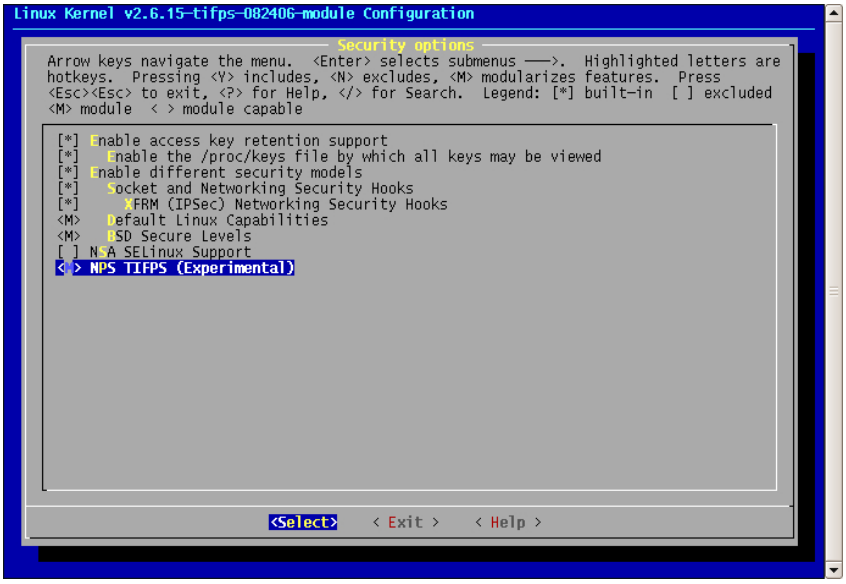


Figure B-2. Set “NPS TIFPS” as a module

i. Exit the kernel configuration utility and save the configuration when prompted.

6. If desired, edit the *EXTRAVERSION* field in main **Makefile** in the **/usr/src/linux** source code directory to custom name the new kernel.

7. Compile the kernel by running:

   # make all && make modules_install && make install

8. Edit the **/boot/grub/grub.conf** file to boot the newly configured kernel by default by changing the *default* field to 0.

9. Edit **/etc/inittab** file to default to runlevel 3 (multi-user mode without X-windows):

   *id:**3**:initdefault:*

10. Reboot the system:

    # reboot

11. Assuming all went as planned, a kernel should be now running a kernel that supports time-based access control for regular files and directories. If the kernel does not boot properly, it is always possible to reboot the system to the previous working kernel by hitting any key at system startup to get to the grub boot menu as shown in Figure B-3 below:

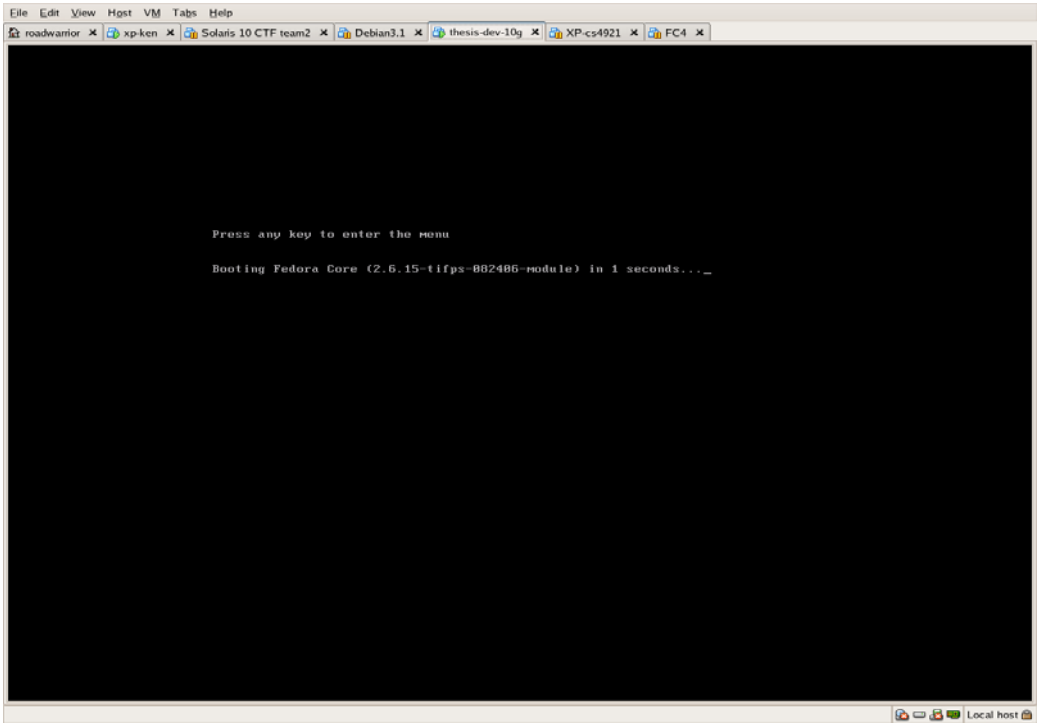


Figure B-3. Fedora Core 5 system start boot screen.

B. INSTALLING THE MODTIME TOOL

1. Install the **modtime** tool and man page for **modtime** by logging in as root and changing into the tifps_tool/ directory on CD1 of archive and running the install script:

 # cd <path to CD1 TIFPS archive/tifps_tool_modtime/

 # ./modtime_install.sh -i

NOTE: to uninstall the tool, run:

 # ./modtime_install.sh -u

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. USERS GUIDE

This appendix describe how to use the TIFPS LSM to control subject and object time permissions as well as the **modtime** tool for interfacing with the time-based access control system.

A. LOADING AND UNLOADING THE TIFPS LSM

Though the TIFPS LSM can be compiled directly into the kernel, it is recommended that it be compiled as a module so that it can be loaded and unloaded dynamically into the kernel by root. After a successful compile and a subsequent reboot into the new kernel, load the TIFPS LSM by running the following command as root:

 # modprobe tifps

To unload the TIFPS LSM run the following command as root:

 # rmmod tifps

It is possible to check whether the TIFPS LSM is loaded into the kernel by listing all the loaded modules. Run the following command as root:

 # lsmod

B. USING THE MODTIME TOOL

The **modtime** tool can be used by the root user to set the persistent time attributes of regular files and directories. It can also be used by user to display the time attribute. It has a simple command line interface similar to other Linux command line tools such as **chmod, chown, ls,** etc… Figure C-1 shows a screen shot for **modtime** tool in use. To get simple usage instructions simply give the command:

 # modtime

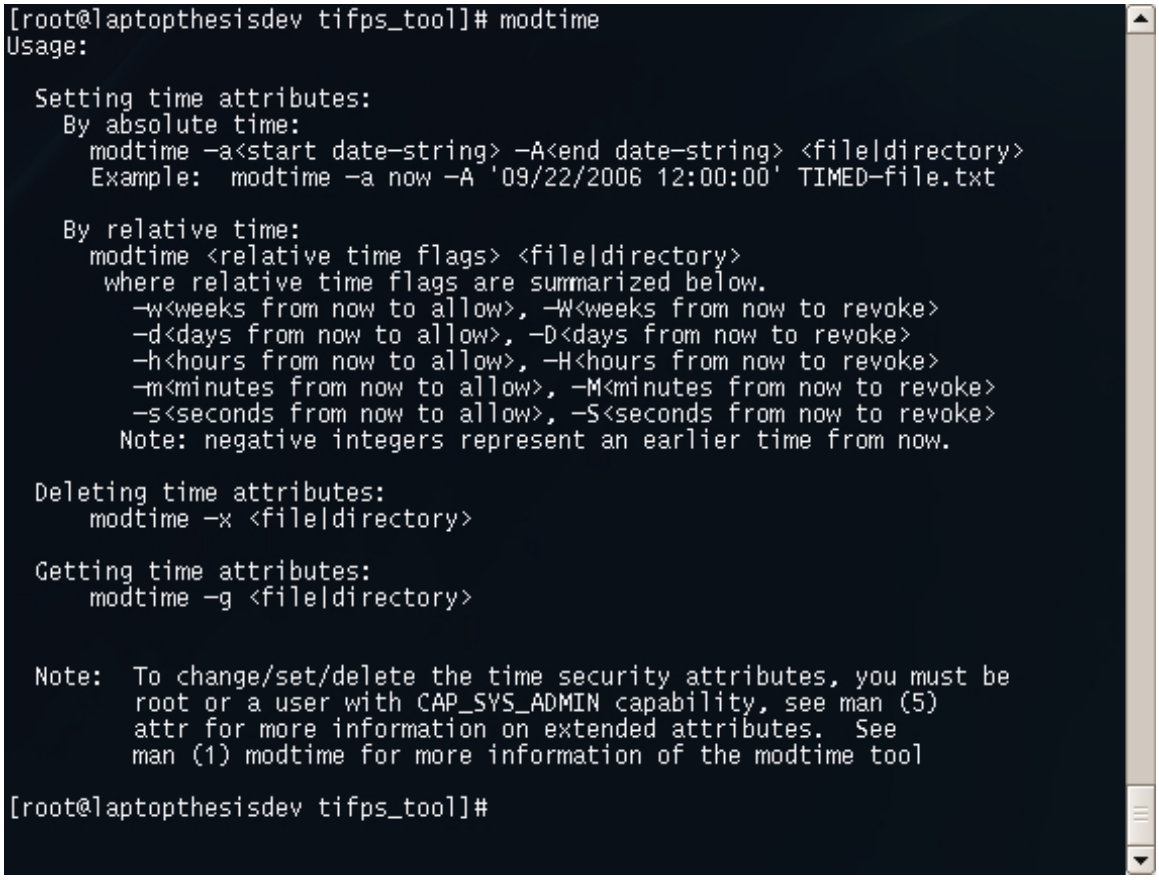


Figure C-1. Screen shot of the command line interface for **modtime**

The **modtime** tool uses flag options and can set time attributes using absolute time or relative time via these flags. It also has flag options for deleting or displaying time attributes. Figure C-2 below shows a screen shot of the man page for modtime. To see the complete man page, type:

 # man modtime

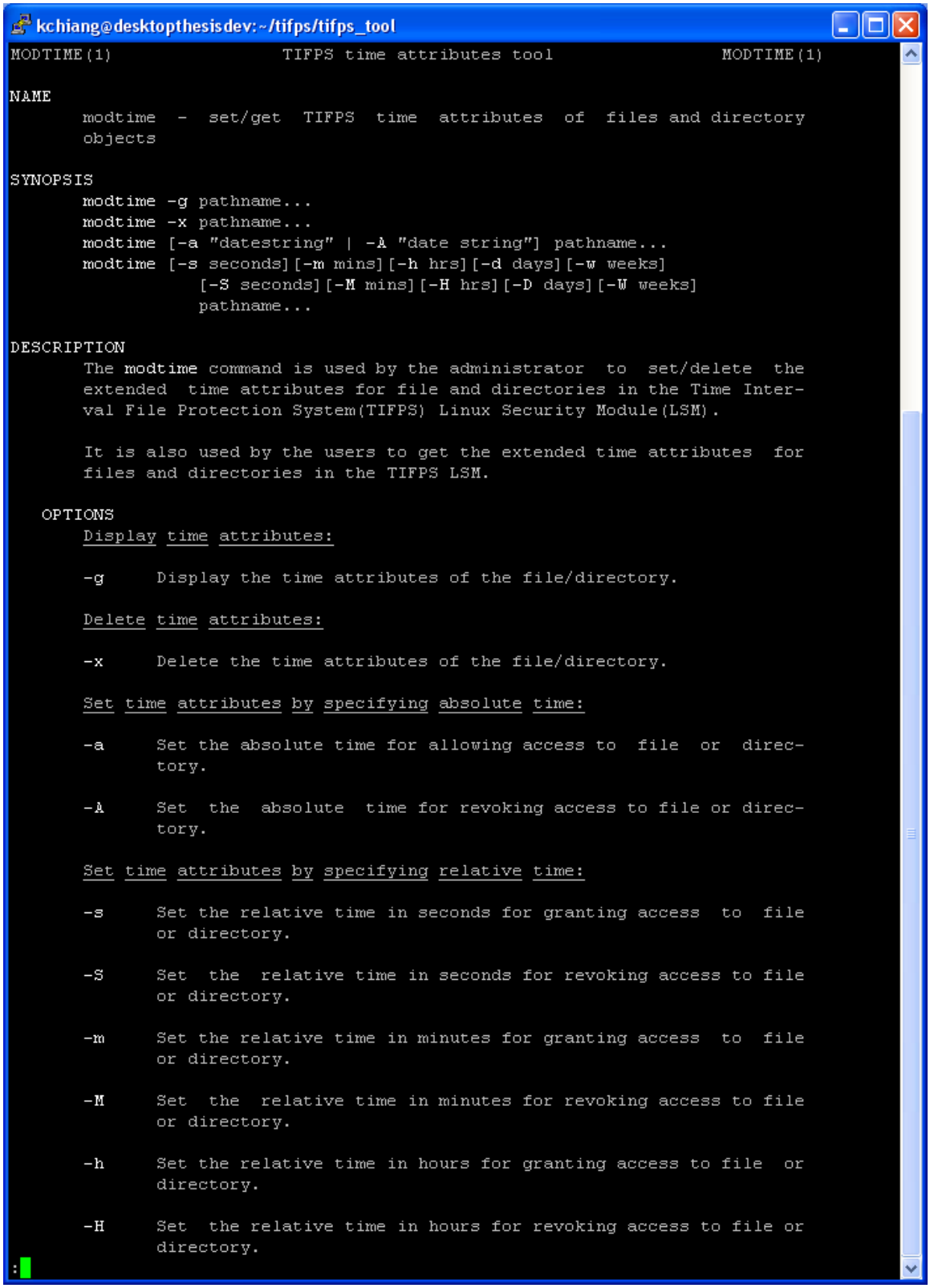


Figure C-2. Screen shot of man page for **modtime**

C. CONTROLLING TIME ATTRIBUTES OF SUBJECTS

To control a user's time interval for allowed access, the super user (root) can set the time attributes of either the **.bash_profile** or **.bashrc** files which reside in the user's home directory. These files are read by **bash** every time a user logs in, therefore, they can be used to set the time attributes of the **bash** shell for the user. For example, to set the time attribute for user ***sam*** to expire on September 22, 2006 at 1700 hrs, run the following:

 # modtime –A '9/22/06 17:00:00' /home/sam/.bash_profile

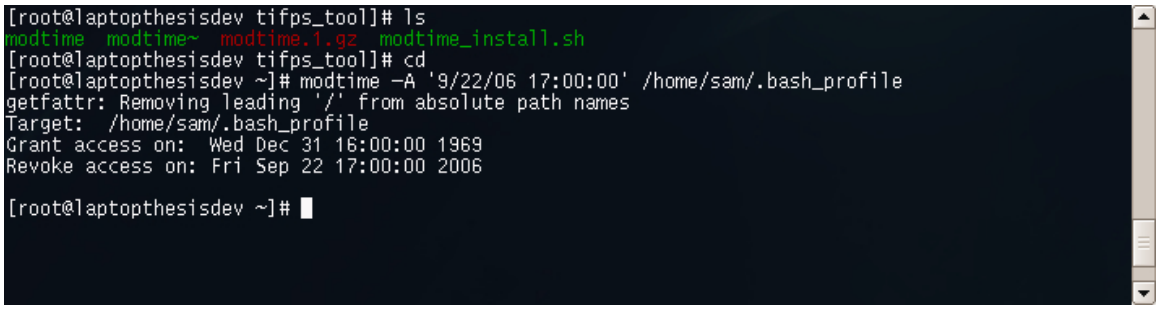


Figure C-3. Screen shot of the **modtime** tool used to set user time attributes.

Since the two **bash** files mentioned above are owned by the user, they can be deleted or moved by the user, effectively bypassing the access control set by root. To prevent this, the root user must also set the file immutable by running the command:

 # chattr +i /home/sam/.bash_profile

To remove time attributes on the **sam** account, run:

 # chattr –i /home/sam/.bash_profile

*# modtime –x /home/sam/.bash**

Note: it is important to use the *–x* flag to remove time attributes for all the **.bash*** files because these files will inherit the attributes set by the administrator during use. For example, the **.bash_history** file is appended each time a user issues a new command in the **bash** shell.

D. CONTROLLING TIME ATTRIBUTES OF OBJECTS

To control access to regular file and directory objects, the **modtime** tool is also used exactly as it was used to set the user attributes above. The following command sets the time interval of allowed access from 30 seconds before current time to September 22, 2006 at 0000 hrs for the **/tmp** directory:

 # modtime –s -30 –A 9/22/06 /tmp

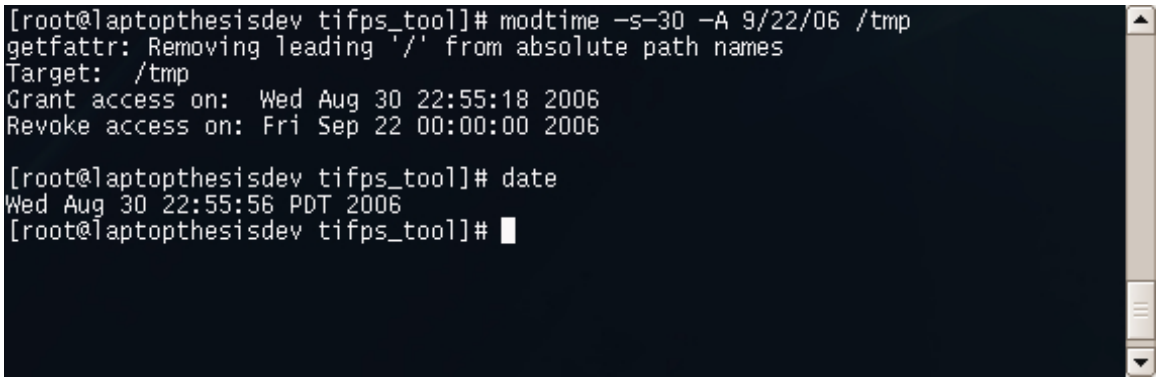


Figure C-4. Screen shot of **modtime** used to control time-based access to **/tmp**

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. TEST PROCEDURES AND RESULTS

This appendix documents the detailed test procedures and results for the test plan described in Chapter IV. Before beginning any testing, ensure that the following preconditions are met:

- The TIFPS LSM is compiled and installed per installation instruction in Appendix B and loaded per usage instructions in Appendix C.
- The **modtime** tool is installed per instructions in Appendix B.
- The individual conducting the tests is logged in as ***root.***
- The following user accounts exist: ***jody, sam,*** and ***don.***
  - Users can be added with the command:

    # useradd – m <username>

  - Set the password by:

    # passwd <username>
- Make a copy of the **testscript/** directory from archive CD 1 to a directory of choice, for example:

  # cp <path to testscript/ directory on CD1> /root

A. ACCESS CONTROL TEST PROCEDURES

Static tests – enforcement of file and directory read/write/execute

1. Navigate to the directory where the TIFPS test scripts are located.

   # cd <path to TIFPS testscripts directory>/accesscontrol

2. Run each of the scripts listed in Table D-1 using the following example format:

   # ./s-read-file-1.sh |tee s-read-file-1-results.txt

3. Compare the results for time intervals t1 to t5 in the resulting text file from each run to Table D-1.

Table D-1. Summary of results expected for each test case *

Test ID	Scripts	Scenario	Expected Results				
			1	2	3	4	5
A1	s-read-file-1.sh to s-read-file-7.sh	1	D	D	D	D	D
A2	s-read-dir-1.sh to s-read-dir-7.sh	2	D	A	D	--	--
A3	s-write-file-1.sh to s-write-file-7.sh	3	D	D	D	D	D
A4	s-write-dir-1.sh to s-write-dir-7.sh	4	D	D	A	D	D
A5	s-exec-file-1.sh to s-exec-file-7.sh	5	D	D	A	D	D
A6	s-exec-dir-1.sh to s-exec-dir-7.sh	6	D	A	D	D	--
A7	s-read-file-6-swap.sh s-write-dir-4-swap.sh	7	D	D	A	D	--

* D = Deny; A = Allow

Static tests- Inheritance in file/directory creation and file copy operations

Inheritance in file and directory creation

1. Run the following commands as ***root***:

   # cd <path to TIFPS directory>/testscripts/accesscontrol/

   # ./s-create-file.sh |tee s-create-file-results.txt

   # ./s-create-dir.sh |tee s-create-dir-results.txt

2. Compare results captured in the results file to expected results summarized in Table D-2.

Table D-2. File and directory creation tests and expected results

Test ID	Test script	Expected Result
B1	s-create-file.sh	Time attributes of the newly created file matches that of the subject.
B2	s-create-dir.sh	Time attributes of the newly created directory matches that of the subject.

Inheritance in file copy

1. Create a login session as ***root*** and clear the time attributes for user ***jody***.

   *# modtime –x /home/jody/.bash**

2. Using the ***root*** session, change to the testscript directory and copy the user scripts to ***jody***'s home directory and make them accessible to the user:

   # cd <path to TIFPS directory>/testscripts/accesscontrol/

   # cp s-copy-file--user.sh /home/jody/*

 # chmod 755 /home/jody/s-copy-file--user.sh*

3. Create a file to capture the results:

   # touch /tmp/s-copy-file-1a-results.txt

   # chmod 766 /tmp/s-copy-file-1a-results.txt

4. Using the ***root*** login session, setup the test case and capture output in the results file:

   # ./s-copy-file-1-admin.sh >> /tmp/s-copy-file-1a-results.txt

5. Create another login session as user ***jody***, run the test case and capture the output in the results file:

   $ ./s-copy-file-1a-user.sh >> /tmp/s-copy-file-1a-results.txt

6. Using the ***root*** login session, cleanup the time attributes for ***jody***:

   *# modtime –x /home/jody/.bash**

7. Repeat steps 4 -6 nine more times for a total of 10 trials. Note: you should logout of the ***jody*** session after each trial and relogin to reinherit the time attributes for ***jody.*** This is especially important for the tests in scenario three. After completing the 10 trials, view the resulting file and ensure that for each trial, the destination file properly inherited the attributes. It should take on the time attributes of the smallest time interval of the three test entities: subject, source object, destination object (See Table D-4 for summary of expected results). Record the number of unsuccessful trials.

8. Repeat steps 3 - 6 using results file, admin script, and user script summarized in Table D-3 below (Test ID C1 is already completed by above):

Table D-3. Summary of results file, admin script, and user script for copy test cases

Test ID	Test Case	Results file	Admin script	User script
C1	Scenario 1, **cp**	s-copy-file-1a-results.txt	s-copy-file-1-admin.sh	s-copy-file-1a-user.sh
C2	Scenario 1, redirection	s-copy-file-1b-results.txt	s-copy-file-1-admin.sh	s-copy-file-1b-user.sh
C3	Scenario1 , pipes	s-copy-file-1c-results.txt	s-copy-file-1-admin.sh	s-copy-file-1c-user.sh
C4	Scenario 2, **cp**	s-copy-file-2a-results.txt	s-copy-file-2-admin.sh	s-copy-file-2a-user.sh
C5	Scenario 2, redirection	s-copy-file-2b-results.txt	s-copy-file-2-admin.sh	s-copy-file-2b-user.sh
C6	Scenario 2, pipes	s-copy-file-2c-results.txt	s-copy-file-2-admin.sh	s-copy-file-2c-user.sh
C7	Scenario 3, **cp**	s-copy-file-3a-results.txt	s-copy-file-3-admin.sh	s-copy-file-3a-user.sh*
C8	Scenario 3, redirection	s-copy-file-3b-results.txt	s-copy-file-3-admin.sh	s-copy-file-3b-user.sh*
C9	Scenario 3, pipes	s-copy-file-3c-results.txt	s-copy-file-3-admin.sh	s-copy-file-3c-user.sh*

* Note: before running these scripts, you must relogin as user jody to reinherit proper time attributes.

Table D-4. Expected results of the file copy tests and file/directory creation tests

Test ID	Script	Expected time attributes of the destination file
C1 – C3	s-copy-file-1(abc)-results.sh	Time attributes of /tmp/dest.txt matches that of destination object
C4 – C6	s-copy-file-2(abc)-results.sh	Time attributes of /tmp/dest.txt matches that of the source object
C7 – C9	s-copy-file-3(abc)-results.sh	Time attributes of /tmp/dest.txt matches that of the subject

Static tests – TIFPS behavior on time expiration during file write operations

This set of tests captures the TIFPS system behavior when access to objects is revoked during a write operation. The scripts attempt to write 5 million 'G's to a file that expires within seconds. Test cases for 1, 2, 3, 4, and 5 seconds are suggested, however, actual number of seconds is dependent on the speed of the hardware running the TIFPS LSM.

1. As root, navigate to the **testscript/accesscontrol/** directory.

 # cd <path to TIFPS directory>/testscripts/accesscontrol/

2. Compile the helper C program used to generate and write 5 million 'G's to the test file.

 # gcc fileprint5M.c –o fileprint5M

3. Run the following script using arguments 1, 2, 3, 4, and 5 or until all 'G's are successfully written to the file **/tmp/write-expired.txt.** The number of characters written successfully to the file will be printed to the screen. See Table D-5 for a sample table used for capturing the information for this test.

 # ./ s-write-expire.sh <number of seconds before access revocation>

Table D-5. Sample table for information to be captured for the access revocation during file write tests

Test ID	Script usage	Number of bytes written successfully out of 5 million	Error Message, if any
D1	s-write-expire.sh 1	*Record # of bytes written*	*Record error message here.*
	s-write-expire.sh 2	*Record # of bytes written*	*Record error message here.*
	s-write-expire.sh 3	*Record # of bytes written*	*Record error message here.*
	s-write-expire.sh 4	*Record # of bytes written*	*Record error message here.*
	s-write-expire.sh 5	*Record # of bytes written*	*Record error message here.*

4. Record the number of characters written successfully and the error message from the system for each test case. Increment the number of seconds until all 5 million 'G's are successfully written to the file and no error message occurs.

Dynamic tests – Dynamically changing subject and object attributes

This set of tests capture system behavior when an administrator changes the time attributes of subjects or objects dynamically while a user is logged into the system. For dynamically changing the subject time attributes case:

1. Create two separate login sessions, one as ***root*** and the other as user ***jody***.
2. In the ***root*** login session, change to the **testscripts/accesscontrol** directory and copy the user scripts to the **/home/jody/** directory and make them accessible to the user:

   # cd <path to TIFPS testscript directory>/accesscontrol/

   # cp d-change--user.sh /home/jody/*

   # chmod 755 /home/jody/d-change--user.sh*
3. In the ***root*** login session, run:

   # ./d-change-subj-admin.sh
4. Immediately (within 10 seconds), run the following script in the ***jody*** login session:

   $ ./d-change-subj-user.sh

5. The expected behavior is continued user access to **/tmp/longfile.txt** because subject time attributes are inherited at user login.

6. For dynamic object changes, repeat steps 3 and 4 as follows:

    a. In the ***root*** login session, run:

    # ./d-change-obj-admin.sh

    b. In the ***jody*** login session, run:

    $ ./d-change-obj-user.sh

7. The expected behavior is revocation of access because object access is checked at every file or directory read/write/execute operation.

Table D-6. Summary of expectations for dynamically changing subject and object time

Test ID	Test scripts	Expected Results
E1	d-change-subj-admin.sh d-change-subj-user.sh	Continued access should be allowed since time attributes are inherited at user login.
E2	d-change-obj-admin.sh d-change-obj-user.sh	Access should be revoked according to the newly set time attributes.

B. ACCESS CONTROL TEST SCRIPTS

This section contains the scripts for the tests described in Section A.

Static tests – enforcement of file and directory read/write/execute

s-read-file-1.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
# Also, make sure the user jody exists or create one.

echo “Static, read file test, scenario 1 of 7”
echo “setup subject time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
echo “”
echo “setup object time attributes...”
echo “this message will self destruct in 10 seconds...” >/tmp/message.txt
modtime -s30 -S40 /tmp/message.txt

sleep 2s

i=1

while [ $i -le 5 ]; do
    date
```

```
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read file test scenario 1 of 7...”
echo “”
```

s-read-file-2.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 2 of 7”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile /tmp/message.txt
echo “”

i=1

while [ $i -le 3 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “Done with read file test, scenario 2 of 7”
echo “”
```

s-read-file-3.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 3 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s20 -S30 /tmp/message.txt

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read file test, scenario 3 of 7”
```

```
echo “”
```

s-read-file-4.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 4 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S30 /home/jody/.bash_profile
modtime -s20 -S40 /tmp/message.txt

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read file test, scenario 4 of 7.”
echo “”
```

s-read-file-5.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 5 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S40 /tmp/message.txt

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done


echo “done static read file test, scenario 5 of 7...”
echo “”
```

s-read-file-6.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 6 of 7”
echo “”
```

```
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s10 -S30 /tmp/message.txt

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done read file test, scenario 6 of 7 ...”
echo “”
```

s-read-file-7.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 7 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S30 /tmp/message.txt

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read file test, scenario 7 of 7...”
echo “”
```

s-read-dir-1.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 1 of 7”
echo “setup subject time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
echo “”
echo “setup object time attributes...”
modtime -s30 -S40 /tmp

sleep 2s

i=1

while [ $i -le 5 ]; do
```

```
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read directory test, scenario 1 of 7”
echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-dir-2.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 2 of 7”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile /tmp
echo “”

sleep 2s

i=1

while [ $i -le 3 ]; do
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read directory test, scenario 2 of 7.”
echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-dir-3.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 3 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s20 -S30 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
```

```
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read directory test, scenario 3 of 7”
echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-dir-4.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 4 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S30 /home/jody/.bash_profile
modtime -s20 -S40 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read directory test, scenario 4 of 7.”
echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-dir-5.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 5 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S40 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
```

```
echo “ done static read directory test, scenario 5 of 7.”

echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-dir-6.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 6 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s10 -S30 /tmp

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read directory test, scenario 6 of 7”
echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-dir-7.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read directory test, scenario 7 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S30 /tmp

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “ls /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static read directory test, scenario 7 of 7”
echo “”
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-write-file-1.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 1 of 7”
echo “setup subject time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
echo “”
echo “create new object and setup its time attributes...”
echo “” > /tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s30 -S40 /tmp/written.txt

sleep 2s

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “echo 'overwritten' >>/tmp/written.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt...”
cat /tmp/written.txt
```

s-write-file-2.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 2 of 7”
echo “setup subject and object time attributes...”
echo “” >/tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s10 -S20 /home/jody/.bash_profile /tmp/written.txt
echo “”

i=1

while [ $i -le 3 ]; do
    date
    su - jody -c “echo 'overwritten `date`' >/tmp/written.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt...”
cat /tmp/written.txt
```

s-write-file-3.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
```

```
echo “Static, file write test, scenario 3 of 7”
echo “”
echo “setup subject and object time attributes...”
echo “” >/tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s20 -S30 /tmp/written.txt

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “echo 'overwritten `date`' >>/tmp/written.txt”

    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt ...”
cat /tmp/written.txt
```

s-write-file-4.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 4 of 7”
echo “”
echo “setup subject and object time attributes...”
echo “”>/tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s10 -S30 /home/jody/.bash_profile
modtime -s20 -S40 /tmp/written.txt

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “echo 'overwritten `date`'>>/tmp/written.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt...”
cat /tmp/written.txt
```

s-write-file-5.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 5 of 7”
echo “”
echo “setup subject and object time attributes...”
echo “” >/tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s20 -S30 /home/jody/.bash_profile
```

```
modtime -s10 -S40 /tmp/written.txt

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “echo 'overwritten `date`'>>/tmp/written.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt...”
cat /tmp/written.txt
```

s-write-file-6.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 6 of 7”
echo “”
echo “setup subject and object time attributes...”
echo “” >/tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s10 -S30 /tmp/written.txt

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “echo 'overwritten `date`'>>/tmp/written.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt...”
cat /tmp/written.txt
```

s-write-file-7.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 7 of 7”
echo “”
echo “setup subject and object time attributes...”
echo “”>/tmp/written.txt
chmod 666 /tmp/written.txt
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S30 /tmp/written.txt

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “echo 'overwritten `date`'>>/tmp/written.txt”
```

```
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp/written.txt...”
cat /tmp/written.txt
```

s-write-dir-1.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory write test, scenario 1 of 7”
echo “setup subject time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
echo “”
echo “”
echo “setup object time attributes...”
modtime -s30 -S40 /tmp

sleep 2s

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-write-dir-2.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, direcotry write test, scenario 2 of 7”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile /tmp
echo “”

i=1

while [ $i -le 3 ]; do
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
```

```
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp diretory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-write-dir-3.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory write test, scenario 3 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s20 -S30 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “touch /tmp`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-write-dir-4.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory write test, scenario 4 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S30 /home/jody/.bash_profile
modtime -s20 -S40 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
```

```
modtime –x /home/jody/.bash* /tmp
```

s-write-dir-5.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory write test, scenario 5 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S40 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-write-dir-6.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory write test, scenario 6 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s10 -S30 /tmp

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-write-dir-7.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
```

```
echo “Static, directory write test, scenario 7 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S30 /tmp

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-file-1.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file execute test, scenario 1 of 7”
echo “setup subject time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
echo “”
echo “setup object time attributes...”
modtime -s30 -S40 /usr/bin/cal

sleep 2s

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-file-2.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file execute test, scenario 2 of 7”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile /usr/bin/cal

i=1
```

```
while [ $i -le 3 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-file-3.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file execute test, scenario 3 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s20 -S30 /usr/bin/cal

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-file-4.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file execute test, scenario 4 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S30 /home/jody/.bash_profile
modtime -s20 -S40 /usr/bin/cal

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
echo “done static exec file test, scenario 4 of 7.”
echo “”
```

```
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-file-5.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 5 of 7”
echo ““
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S40 /usr/bin/cal

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo ““
    sleep 10s
    let “i=$i+1”
done

echo “done static exec file test, scenario 5 of 7.”
echo ““
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-file-6.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, file write test, scenario 6 of 7”
echo ““
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s10 -S30 /usr/bin/cal

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo ““
    sleep 10s
    let “i=$i+1”
done
echo “done exec file test, scenario 6 of 7”
echo ““
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-file-7.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
```

```
echo “Static, file execute test, scenario 7 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S30 /usr/bin/cal

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cal”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done static exec file test, scenario 7 of 7.”
echo “”
#cleanup
modtime –x /home/jody/.bash* /usr/bin/cal
```

s-exec-dir-1.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 1 of 7”
echo “setup subject time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
echo “”
echo “setup object time attributes...”
modtime -s30 -S40 /tmp

sleep 2s

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-dir-2.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 2 of 7”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile /tmp

sleep 2s
```

```
i=1

while [ $i -le 3 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-dir-3.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 3 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s20 -S30 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-dir-4.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 4 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S30 /home/jody/.bash_profile
modtime -s20 -S40 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
```

```
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-dir-5.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 5 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S40 /tmp

i=1

while [ $i -le 5 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-dir-6.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 6 of 7”
echo “”
echo “setup subject and object time attributes...”
modtime -s10 -S20 /home/jody/.bash_profile
modtime -s10 -S30 /tmp

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-exec-dir-7.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory execute test, scenario 7 of 7”
echo “”
echo “setup subject and object time attributes...”
```

```
modtime -s20 -S30 /home/jody/.bash_profile
modtime -s10 -S30 /tmp

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cd /tmp”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done
#cleanup
modtime –x /home/jody/.bash* /tmp
```

s-read-file-6-swap.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, read file test, scenario 6 of 7(subject and object time swapped”
echo “”
echo “setup subject and object time attributes...”
#modtime -s10 -S20 /home/jody/.bash_profile
#modtime -s10 -S30 /tmp/message.txt
modtime -s10 –S30 /home/jody/.bash_profile
modtime -s10 –S20 /tmp/message.txt

i=1

while [ $i -le 4 ]; do
    date
    su - jody -c “cat /tmp/message.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo “”
    sleep 10s
    let “i=$i+1”
done

echo “done read file test, scenario 6 of 7 ...”
echo “”
```

s-write-dir-4-swap.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, directory write test, scenario 4 of 7 (subject and object swapped)”
echo “”
echo “setup subject and object time attributes...”
#modtime -s10 -S30 /home/jody/.bash_profile
#modtime -s20 -S40 /tmp
modtime –s20 –S40 /home/jody/.bash_profile
modtime –s10 –S30 /tmp

i=1

while [ $i -le 5 ]; do
```

```
    date
    su - jody -c “touch /tmp/`date +%T`-$i.txt”
    if [ $? -gt 0 ]; then
        echo “t$i: Access Denied”
    else
        echo “t$i: Access Granted”
    fi
    echo ““
    sleep 10s
    let “i=$i+1”
done

echo “contents of /tmp directory”
ls /tmp
#cleanup
modtime –x /home/jody/.bash* /tmp
```

Static tests- Inheritance in file/directory creation and file copy operations

s-create-file.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, create file test”
echo ““
echo ““
echo “setup subject time attributes...”
modtime -w-2 -W2 /home/jody/.bash_profile
echo ““
echo ““

echo “current time is:”
date
echo ““

echo “creating a new file...........”
su - jody -c “echo 'new file from jody' > jodynew.txt”
echo ““

echo “The time attribute for the newly created file is ...”
modtime -g /home/jody/jodynew.txt
rm -f /home/jody/jodynew.txt
```

s-create-dir.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script

echo “Static, create directory test”
echo ““
echo ““
echo “setup subject time attributes...”
modtime -w-2 -W2 /home/jody/.bash_profile
echo ““
echo ““

echo “current time is:”
date
echo ““

echo “creating a new directory...........”
su - jody -c “mkdir jodyNewDirectory”
echo ““

echo “The time attribute for the newly created directory is ...”
modtime -g /home/jody/jodyNewDirectory
rm -rf /home/jody/jodyNewDirectory
```

s-copy-file-1-admin.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
echo “setup subject time attributes...”
cd /home/jody/
modtime –x /home/jody/.bash*
modtime -w-2 -W2 /home/jody/.bash_profile
echo “setup source object time attributes...”
cd /tmp
echo “This is the source file.” >/tmp/source.txt
modtime -d-1 -D1 source.txt
echo “setup destination object time attributes...(smallest)”
echo “This is the destination file.” >/tmp/dest.txt
chmod 777 /tmp/dest.txt /tmp/source.txt
modtime -h-1 -H1 dest.txt
```

s-copy-file-1a-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-1-admin.sh script as root first.
echo “Static, copy file test (using cp), scenario 1 of 3 - smallest dest object”
echo “”
echo “current time is:”
date
cd /tmp
cp source.txt dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-1b-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-1-admin.sh script as root first.
echo “Static, copy file test (using redirection), scenario 1 of 3 - smallest dest object”
echo “”
echo “current time is:”
date
cd /tmp
cat source.txt > dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-1c-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-1-admin.sh script as root first.
echo “Static, copy file test (using pipes), scenario 1 of 3 - smallest dest obj”
echo “”
echo “current time is:”
date
cd /tmp
cat source.txt |tee dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-2-admin.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
echo “setup subject time attributes...”
cd /home/jody/
modtime –x /home/jody/.bash*
modtime -w-2 -W2 /home/jody/.bash_profile
```

```
echo “setup source object time attributes... (smallest)”
cd /tmp
echo “This is the source file.” >/tmp/source.txt
modtime -h-1 -H1 source.txt
echo “setup destination object time attributes...”
echo “This is the destination file.” >/tmp/dest.txt
chmod 777 /tmp/dest.txt /tmp/source.txt
modtime -d-1 -D1 dest.txt
```

s-copy-file-2a-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-2-admin.sh script as root first.
echo “Static, copy file test (using cp), scenario 2 of 3 - smallest src object”
echo “”
echo “current time is:”
date
cd /tmp
cp source.txt dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-2b-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-2-admin.sh script as root first.
echo “Static, copy file test (using redirection), scenario 2 of 3 - smallest src object”
echo “”
echo “current time is:”
date
cd /tmp
cat source.txt > dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-2c-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-2-admin.sh script as root first.
echo “Static, copy file test (using pipes), scenario 2 of 3 - smallest src object”
echo “”
echo “current time is:”
date
cd /tmp
cat source.txt |tee dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-3-admin.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
echo “setup subject time attributes...(smallest)”
cd /home/jody/
modtime –x /home/jody/.bash*
modtime -h-1 -H1 /home/jody/.bash_profile
echo “setup source object time attributes...”
cd /tmp
echo “This is the source file.” >/tmp/source.txt
modtime -d-1 -D1 source.txt
echo “setup destination object time attributes...”
echo “This is the destination file.” >/tmp/dest.txt
chmod 777 /tmp/dest.txt /tmp/source.txt
```

```
modtime -d-1 -D1 dest.txt
```

s-copy-file-3a-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-3-admin.sh script as root first.
echo “Static, copy file test (using cp), scenario 3 of 3 - smallest subject”
echo “”
echo “current time is:”
date
cd /tmp
cp source.txt dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-3b-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-3-admin.sh script as root first.
echo “Static, copy file test (using redirection), scenario 3 of 3 - smallest subject”
echo “”
echo “current time is:”
date
cd /tmp
cat source.txt > dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

s-copy-file-3c-user.sh

```
#!/bin/bash
# setup the test by running the s-copy-file-3-admin.sh script as root first.
echo “Static, copy file test (using pipes), scenario 3 of 3 - smallest subject”
echo “”
echo “current time is:”
date
cd /tmp
cat source.txt |tee dest.txt
echo “The resulting time attribute for the destination file is ...”
modtime -g dest.txt
cd
```

Static tests – TIFPS behavior on time expiration during file write operations

fileprint5M.c

```
#include <stdio.h>

int main(){
        FILE *fp;
        fp = fopen(“/tmp/write-expired.txt”, “r+”);

        if (fp ==NULL){
                printf(“ERROR opening file:  goodbye!\n”);
                return 0;
        }

        int i=0;
        int err;
        for (i=0; i<5000000; i++){
                err=fprintf(fp, “G”);
                if (err < 0){
```

```
                        printf("ERROR writing to file: ERR %d\n", err);
                        return 0;
                }
        }
        err = fclose(fp);
        if (err){
                printf("ERROR closing file:  ERR %d\n", err);
                return 0;
        }
        printf("File write successfully completed!\n");
        return 0;
}
```

s-write-expire.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
# give the number of seconds to revoke access as the first argument

if [ $EUID -gt 0 ];then
    echo "this script must be run as root"
    exit;
fi

if [ $# -lt 1 ]; then
    echo "Give the number of seconds before access revocation as a first argument"
    exit;
fi

echo "Static test: File expiration during write operation"
echo ""
echo "setup subject and object time attributes..."
echo "file will expire in $1 second(s)"
modtime -W 1 /home/jody/.bash_profile
rm -f /tmp/write-expired.txt
touch /tmp/write-expired.txt
chmod 777 /tmp/write-expired.txt
modtime -S $1 /tmp/write-expired.txt

echo "write operation started:  "
date

echo ""
echo "User jody tries to append 5 million G's to /tmp/write-expired.txt file .."
su jody -c "./fileprint5M"
echo ""
echo "write operation ended:"
date

echo "Number of characters written to the file successfully:"
wc -c /tmp/write-expired.txt
echo ""
```

Dynamic tests – Dynamically changing subject and object attributes

d-change-subj-admin.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
# Run this script as root; while the script sleeps, login as jody and run the
# d-change-subj-user.sh script.
echo "Dynamic test, change subject attributes while user is logged in and reading a file"
echo ""
echo ""
echo "Initialize subject time attributes..."
modtime -w-2 -W2 /home/jody/.bash_profile
echo ""
echo ""
```

```
echo “setup object and its time attributes...”
rm /tmp/longfile.txt
i=1
while [ $i -le 20 ]; do
    echo “line $i: This is a long file” >> /tmp/longfile.txt
    let “i=$i+1”
done
modtime -w-1 -W1 /tmp/longfile.txt

echo “current time is:”
date
echo ““

echo “going to sleep for 10s...”
sleep 10s
echo “...”
echo “...”
echo ““

echo “current time is: “
date
echo ““
echo “changing subject time attributes...”
modtime -S-1 /home/jody/.bash_profile
```

d-change-subj-user.sh

```
#!/bin/bash
# This script is a companion to the d-change-subj-admin.sh script;
# Run this script as user jody as soon as the main script sleeps

echo “current time is:”
date
echo ““

cat /tmp/longfile.txt

echo “sleeping for 10s....”
sleep 10s
echo “.............................”
echo ““
echo “current time is:”
date
echo ““
cat /tmp/longfile.txt
```

d-change-obj-admin.sh

```
#!/bin/bash
# must be run as root, be sure tifps LSM is loaded before running script
# Run this script as root; while the script sleeps, login as jody and run the
# d-change-obj-user.sh script.
echo “Dynamic test, change object attributes while user is logged in and reading the
object”
echo ““
echo ““
echo “Initialize subject time attributes...”
modtime -w-2 -W2 /home/jody/.bash_profile
echo ““
echo ““
echo “setup object and its time attributes...”
rm /tmp/longfile.txt
i=1
while [ $i -le 20 ]; do
    echo “line $i: This is a long file” >> /tmp/longfile.txt
    let “i=$i+1”
done
modtime -w-1 -W1 /tmp/longfile.txt
```

```
echo “current time is:”
date
echo ““

echo “going to sleep for 10s...”
sleep 10s
echo “...”
echo “...”
echo ““

echo “current time is: “
date
echo ““
echo “changing object time attributes...”
modtime -S-1 /tmp/longfile.txt
```

d-change-obj-user.sh

```
#!/bin/bash
# This script is a companion to the d-change-obj-admin.sh script;
# Run this script as user jody as soon as the main script sleeps

echo “current time is:”
date
echo ““

cat /tmp/longfile.txt

echo “sleeping for 10s....”
sleep 10s
echo “.............................”
echo ““
echo “current time is:”
date
echo ““
cat /tmp/longfile.txt
```

C. ACCESS CONTROL TEST RESULTS

This section contains the raw test results for the tests described in Section A. The dates, times, and contents of directories displayed here will be slightly different compared to new test results obtained by the tester.

Static tests – enforcement of file and directory read/write/execute

s-read-file--results.txt*

```
Static, read file test, scenario 1 of 7
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 20:33:43 2006
Revoke access on: Mon Sep  4 20:33:53 2006


setup object time attributes...
Target:  /tmp/message.txt
Grant access on:  Mon Sep  4 20:34:03 2006
Revoke access on: Mon Sep  4 20:34:13 2006

Mon Sep  4 20:33:36 PDT 2006
t1: Access Denied
```

```
Mon Sep  4 20:33:47 PDT 2006
t2: Access Denied

Mon Sep  4 20:33:59 PDT 2006
t3: Access Denied

Mon Sep  4 20:34:10 PDT 2006
t4: Access Denied

Mon Sep  4 20:34:21 PDT 2006
t5: Access Denied

done static read file test scenario 1 of 7...

Static, read file test, scenario 2 of 7
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 20:42:17 2006
Revoke access on: Mon Sep  4 20:42:27 2006

Target:  /tmp/message.txt
Grant access on:  Mon Sep  4 20:42:17 2006
Revoke access on: Mon Sep  4 20:42:27 2006


Mon Sep  4 20:42:10 PDT 2006
t1: Access Denied

Mon Sep  4 20:42:21 PDT 2006
this message will self destruct in 10 seconds...
t2: Access Granted

Mon Sep  4 20:42:33 PDT 2006
t3: Access Denied

Done with read file test, scenario 2 of 7

Static, read file test, scenario 3 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:27:38 2006
Revoke access on: Mon Sep  4 21:27:48 2006

Target:  /tmp/message.txt
Grant access on:  Mon Sep  4 21:27:48 2006
Revoke access on: Mon Sep  4 21:27:58 2006

Mon Sep  4 21:27:29 PDT 2006
t1: Access Denied

Mon Sep  4 21:27:40 PDT 2006
t2: Access Denied

Mon Sep  4 21:27:52 PDT 2006
t3: Access Denied
```

```
Mon Sep  4 21:28:03 PDT 2006
t4: Access Denied

Mon Sep  4 21:28:14 PDT 2006
t5: Access Denied

done static read file test, scenario 3 of 7

Static, read file test, scenario 4 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Tue Sep  5 19:27:16 2006
Revoke access on: Tue Sep  5 19:27:36 2006

/tmp/message.txt does not currently have accessible TIFPS attributes
Target:  /tmp/message.txt
Grant access on:  Tue Sep  5 19:27:26 2006
Revoke access on: Tue Sep  5 19:27:46 2006

Tue Sep  5 19:27:06 PDT 2006
t1: Access Denied

Tue Sep  5 19:27:17 PDT 2006
t2: Access Denied

Tue Sep  5 19:27:27 PDT 2006
this message will self destruct in 10 seconds...
t3: Access Granted

Tue Sep  5 19:27:38 PDT 2006
t4: Access Denied

Tue Sep  5 19:27:49 PDT 2006
t5: Access Denied

done static read file test, scenario 4 of 7.


setup subject and object time attributes...
Static, read file test, scenario 5 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 20:56:50 2006
Revoke access on: Mon Sep  4 20:57:00 2006

Target:  /tmp/message.txt
Grant access on:  Mon Sep  4 20:56:41 2006
Revoke access on: Mon Sep  4 20:57:11 2006

Mon Sep  4 20:56:31 PDT 2006
t1: Access Denied

Mon Sep  4 20:56:42 PDT 2006
t2: Access Denied
```

```
Mon Sep  4 20:56:53 PDT 2006
this message will self destruct in 10 seconds...
t3: Access Granted

Mon Sep  4 20:57:04 PDT 2006
t4: Access Denied

Mon Sep  4 20:57:15 PDT 2006
t5: Access Denied

done static read file test, scenario 5 of 7...

Static, read file test, scenario 6 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 20:59:52 2006
Revoke access on: Mon Sep  4 21:00:02 2006

Target:  /tmp/message.txt
Grant access on:  Mon Sep  4 20:59:52 2006
Revoke access on: Mon Sep  4 21:00:12 2006

Mon Sep  4 20:59:43 PDT 2006
t1: Access Denied

Mon Sep  4 20:59:54 PDT 2006
this message will self destruct in 10 seconds...
t2: Access Granted

Mon Sep  4 21:00:05 PDT 2006
t3: Access Denied

Mon Sep  4 21:00:16 PDT 2006
t4: Access Denied

done read file test, scenario 6 of 7 ...


Static, read file test, scenario 7 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:01:47 2006
Revoke access on: Mon Sep  4 21:01:57 2006

Target:  /tmp/message.txt
Grant access on:  Mon Sep  4 21:01:37 2006
Revoke access on: Mon Sep  4 21:01:57 2006

Mon Sep  4 21:01:27 PDT 2006
t1: Access Denied

Mon Sep  4 21:01:38 PDT 2006
t2: Access Denied
```

```
Mon Sep  4 21:01:49 PDT 2006
this message will self destruct in 10 seconds...
t3: Access Granted

Mon Sep  4 21:02:01 PDT 2006
t4: Access Denied

done static read file test, scenario 7 of 7...
```

s-read-dir--results.txt*

```
Static, read directory test, scenario 1 of 7
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:24:12 2006
Revoke access on: Mon Sep  4 22:24:22 2006


setup object time attributes...
/tmp does not currently have accessible TIFPS attributes
Target:  /tmp
Grant access on:  Mon Sep  4 22:24:32 2006
Revoke access on: Mon Sep  4 22:24:42 2006

Mon Sep  4 22:24:05 PDT 2006
t1: Access Denied

Mon Sep  4 22:24:17 PDT 2006
t2: Access Denied

Mon Sep  4 22:24:29 PDT 2006
t3: Access Denied

Mon Sep  4 22:24:40 PDT 2006
t4: Access Denied

Mon Sep  4 22:24:51 PDT 2006
t5: Access Denied

done static read directory test, scenario 1 of 7

Static, read directory test, scenario 2 of 7
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:25:21 2006
Revoke access on: Mon Sep  4 22:25:31 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:25:21 2006
Revoke access on: Mon Sep  4 22:25:31 2006


Mon Sep  4 22:25:14 PDT 2006
t1: Access Denied

Mon Sep  4 22:25:25 PDT 2006
06:26:03-2.txt
```

```
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt
t2: Access Granted

Mon Sep  4 22:25:36 PDT 2006
t3: Access Denied

done static read directory test, scenario 2 of 7.

Static, read directory test, scenario 3 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:26:47 2006
Revoke access on: Mon Sep  4 22:26:57 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:26:57 2006
Revoke access on: Mon Sep  4 22:27:07 2006

Mon Sep  4 22:26:37 PDT 2006
t1: Access Denied

Mon Sep  4 22:26:49 PDT 2006
t2: Access Denied

Mon Sep  4 22:27:00 PDT 2006
t3: Access Denied

Mon Sep  4 22:27:11 PDT 2006
t4: Access Denied

Mon Sep  4 22:27:22 PDT 2006
t5: Access Denied

done static read directory test, scenario 3 of 7

Static, read directory test, scenario 4 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:28:30 2006
Revoke access on: Mon Sep  4 22:28:50 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:28:40 2006
Revoke access on: Mon Sep  4 22:29:00 2006
```

```
Mon Sep  4 22:28:20 PDT 2006
t1: Access Denied

Mon Sep  4 22:28:31 PDT 2006
t2: Access Denied

Mon Sep  4 22:28:43 PDT 2006
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt
t3: Access Granted

Mon Sep  4 22:28:54 PDT 2006
t4: Access Denied

Mon Sep  4 22:29:05 PDT 2006
t5: Access Denied

done static read directory test, scenario 4 of 7.

Static, read directory test, scenario 5 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:29:57 2006
Revoke access on: Mon Sep  4 22:30:07 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:29:47 2006
Revoke access on: Mon Sep  4 22:30:17 2006

Mon Sep  4 22:29:38 PDT 2006
t1: Access Denied

Mon Sep  4 22:29:49 PDT 2006
t2: Access Denied

Mon Sep  4 22:30:00 PDT 2006
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
mapping-kchiang
mapping-root
message.txt
```

```
s-read-file-2-results.txt
written.txt
t3: Access Granted

Mon Sep  4 22:30:12 PDT 2006
t4: Access Denied

Mon Sep  4 22:30:23 PDT 2006
t5: Access Denied

 done static read directory test, scenario 5 of 7.


Static, read directory test, scenario 6 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:30:59 2006
Revoke access on: Mon Sep  4 22:31:09 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:31:00 2006
Revoke access on: Mon Sep  4 22:31:20 2006

Mon Sep  4 22:30:50 PDT 2006
t1: Access Denied

Mon Sep  4 22:31:01 PDT 2006
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt
t2: Access Granted

Mon Sep  4 22:31:13 PDT 2006
t3: Access Denied

Mon Sep  4 22:31:24 PDT 2006
t4: Access Denied

done static read directory test, scenario 6 of 7

Static, read directory test, scenario 7 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:33:34 2006
Revoke access on: Mon Sep  4 22:33:44 2006
```

```
Target:  /tmp
Grant access on:  Mon Sep  4 22:33:24 2006
Revoke access on: Mon Sep  4 22:33:44 2006

Mon Sep  4 22:33:14 PDT 2006
t1: Access Denied

Mon Sep  4 22:33:26 PDT 2006
t2: Access Denied

Mon Sep  4 22:33:37 PDT 2006
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt
t3: Access Granted

Mon Sep  4 22:33:49 PDT 2006
t4: Access Denied

done static read directory test, scenario 7 of 7
```

s-write-file--results.txt*

```
Static, file write test, scenario 1 of 7
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:14:37 2006
Revoke access on: Mon Sep  4 21:14:47 2006


create new object and setup its time attributes...
Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:14:58 2006
Revoke access on: Mon Sep  4 21:15:08 2006

Mon Sep  4 21:14:30 PDT 2006
t1: Access Denied

Mon Sep  4 21:14:41 PDT 2006
t2: Access Denied

Mon Sep  4 21:14:52 PDT 2006
t3: Access Denied

Mon Sep  4 21:15:04 PDT 2006
t4: Access Denied
```

```
Mon Sep  4 21:15:15 PDT 2006
t5: Access Denied

contents of /tmp/written.txt...

Static, file write test, scenario 2 of 7
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:16:51 2006
Revoke access on: Mon Sep  4 21:17:01 2006

Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:16:51 2006
Revoke access on: Mon Sep  4 21:17:01 2006


Mon Sep  4 21:16:41 PDT 2006
t1: Access Denied

Mon Sep  4 21:16:52 PDT 2006
t2: Access Granted

Mon Sep  4 21:17:05 PDT 2006
t3: Access Denied

contents of /tmp/written.txt...
overwritten Mon Sep  4 21:16:52 PDT 2006
Static, file write test, scenario 3 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:21:45 2006
Revoke access on: Mon Sep  4 21:21:55 2006

Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:21:55 2006
Revoke access on: Mon Sep  4 21:22:05 2006

Mon Sep  4 21:21:35 PDT 2006
t1: Access Denied

Mon Sep  4 21:21:46 PDT 2006
t2: Access Denied

Mon Sep  4 21:21:58 PDT 2006
t3: Access Denied

Mon Sep  4 21:22:09 PDT 2006
t4: Access Denied

Mon Sep  4 21:22:20 PDT 2006
t5: Access Denied

contents of /tmp/written.txt ...

Static, file write test, scenario 4 of 7
```

```
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:23:06 2006
Revoke access on: Mon Sep  4 21:23:26 2006

Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:23:16 2006
Revoke access on: Mon Sep  4 21:23:36 2006

Mon Sep  4 21:22:56 PDT 2006
t1: Access Denied

Mon Sep  4 21:23:08 PDT 2006
t2: Access Denied

Mon Sep  4 21:23:20 PDT 2006
t3: Access Granted

Mon Sep  4 21:23:33 PDT 2006
t4: Access Denied

Mon Sep  4 21:23:46 PDT 2006
t5: Access Denied

contents of /tmp/written.txt...

overwritten Mon Sep  4 21:23:20 PDT 2006
Static, file write test, scenario 5 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:30:39 2006
Revoke access on: Mon Sep  4 21:30:49 2006

Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:30:30 2006
Revoke access on: Mon Sep  4 21:31:00 2006

Mon Sep  4 21:30:20 PDT 2006
t1: Access Denied

Mon Sep  4 21:30:32 PDT 2006
t2: Access Denied

Mon Sep  4 21:30:43 PDT 2006
t3: Access Granted

Mon Sep  4 21:30:56 PDT 2006
t4: Access Denied

Mon Sep  4 21:31:08 PDT 2006
t5: Access Denied

contents of /tmp/written.txt...

overwritten Mon Sep  4 21:30:44 PDT 2006
Static, file write test, scenario 6 of 7
```

```
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:32:48 2006
Revoke access on: Mon Sep  4 21:32:58 2006

Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:32:48 2006
Revoke access on: Mon Sep  4 21:33:08 2006

Mon Sep  4 21:32:39 PDT 2006
t1: Access Denied

Mon Sep  4 21:32:50 PDT 2006
t2: Access Granted

Mon Sep  4 21:33:02 PDT 2006
t3: Access Denied

Mon Sep  4 21:33:14 PDT 2006
t4: Access Denied

contents of /tmp/written.txt...

overwritten Mon Sep  4 21:32:50 PDT 2006


Static, file write test, scenario 7 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:34:26 2006
Revoke access on: Mon Sep  4 21:34:36 2006

Target:  /tmp/written.txt
Grant access on:  Mon Sep  4 21:34:16 2006
Revoke access on: Mon Sep  4 21:34:36 2006

Mon Sep  4 21:34:07 PDT 2006
t1: Access Denied

Mon Sep  4 21:34:18 PDT 2006
t2: Access Denied

Mon Sep  4 21:34:29 PDT 2006
t3: Access Granted

Mon Sep  4 21:34:41 PDT 2006
t4: Access Denied

contents of /tmp/written.txt...

overwritten Mon Sep  4 21:34:29 PDT 2006
```

s-write-dir--results.txt*

```
Static, directory write test, scenario 1 of 7
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:35:48 2006
Revoke access on: Mon Sep  4 22:35:58 2006


setup object time attributes...
Target:  /tmp
Grant access on:  Mon Sep  4 22:36:08 2006
Revoke access on: Mon Sep  4 22:36:18 2006

Mon Sep  4 22:35:40 PDT 2006
t1: Access Denied

Mon Sep  4 22:35:51 PDT 2006
t2: Access Denied

Mon Sep  4 22:36:03 PDT 2006
t3: Access Denied

Mon Sep  4 22:36:15 PDT 2006
t4: Access Denied

Mon Sep  4 22:36:26 PDT 2006
t5: Access Denied

contents of /tmp directory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt


Static, direcotry write test, scenario 2 of 7
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:38:18 2006
Revoke access on: Mon Sep  4 22:38:28 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:38:18 2006
Revoke access on: Mon Sep  4 22:38:28 2006


Mon Sep  4 22:38:08 PDT 2006
t1: Access Denied
```

```
Mon Sep  4 22:38:20 PDT 2006
t2: Access Granted

Mon Sep  4 22:38:33 PDT 2006
t3: Access Denied

contents of /tmp diretory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
22:38:20-2.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt


Static, directory write test, scenario 3 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:39:21 2006
Revoke access on: Mon Sep  4 22:39:31 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:39:31 2006
Revoke access on: Mon Sep  4 22:39:41 2006

Mon Sep  4 22:39:12 PDT 2006
t1: Access Denied

Mon Sep  4 22:39:24 PDT 2006
t2: Access Denied

Mon Sep  4 22:39:36 PDT 2006
t3: Access Denied

Mon Sep  4 22:39:47 PDT 2006
t4: Access Denied

Mon Sep  4 22:39:58 PDT 2006
t5: Access Denied

contents of /tmp directory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
```

```
22:38:20-2.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt


Static, directory write test, scenario 4 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:42:08 2006
Revoke access on: Mon Sep  4 22:42:28 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:42:18 2006
Revoke access on: Mon Sep  4 22:42:38 2006

Mon Sep  4 22:41:59 PDT 2006
t1: Access Denied

Mon Sep  4 22:42:10 PDT 2006
t2: Access Denied

Mon Sep  4 22:42:21 PDT 2006
t3: Access Granted

Mon Sep  4 22:42:32 PDT 2006
t4: Access Denied

Mon Sep  4 22:42:43 PDT 2006
t5: Access Denied

contents of /tmp directory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
22:38:20-2.txt
22:42:21-3.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt



Static, directory write test, scenario 5 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:44:03 2006
```

```
Revoke access on: Mon Sep  4 22:44:13 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:43:53 2006
Revoke access on: Mon Sep  4 22:44:23 2006

Mon Sep  4 22:43:44 PDT 2006
t1: Access Denied

Mon Sep  4 22:43:55 PDT 2006
t2: Access Denied

Mon Sep  4 22:44:06 PDT 2006
t3: Access Granted

Mon Sep  4 22:44:18 PDT 2006
t4: Access Denied

Mon Sep  4 22:44:29 PDT 2006
t5: Access Denied

contents of /tmp directory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
22:38:20-2.txt
22:42:21-3.txt
22:44:06-3.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt

Static, directory write test, scenario 6 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:45:38 2006
Revoke access on: Mon Sep  4 22:45:48 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:45:38 2006
Revoke access on: Mon Sep  4 22:45:58 2006

Mon Sep  4 22:45:29 PDT 2006
t1: Access Denied

Mon Sep  4 22:45:40 PDT 2006
t2: Access Granted

Mon Sep  4 22:45:52 PDT 2006
t3: Access Denied
```

```
Mon Sep  4 22:46:03 PDT 2006
t4: Access Denied

contents of /tmp directory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
22:38:20-2.txt
22:42:21-3.txt
22:44:06-3.txt
22:45:40-2.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt

Static, directory write test, scenario 7 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:47:44 2006
Revoke access on: Mon Sep  4 22:47:54 2006

Target:  /tmp
Grant access on:  Mon Sep  4 22:47:34 2006
Revoke access on: Mon Sep  4 22:47:54 2006

Mon Sep  4 22:47:24 PDT 2006
t1: Access Denied

Mon Sep  4 22:47:36 PDT 2006
t2: Access Denied

Mon Sep  4 22:47:47 PDT 2006
t3: Access Granted

Mon Sep  4 22:47:58 PDT 2006
t4: Access Denied

contents of /tmp directory
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
22:38:20-2.txt
22:42:21-3.txt
22:44:06-3.txt
22:45:40-2.txt
```

```
22:47:47-3.txt
mapping-kchiang
mapping-root
message.txt
s-read-file-2-results.txt
written.txt
```

s-exec-file--results.txt*

```
Static, file execute test, scenario 1 of 7
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:36:16 2006
Revoke access on: Mon Sep  4 21:36:26 2006


setup object time attributes...
Target:  /usr/bin/cal
Grant access on:  Mon Sep  4 21:36:36 2006
Revoke access on: Mon Sep  4 21:36:46 2006

Mon Sep  4 21:36:09 PDT 2006
t1: Access Denied

Mon Sep  4 21:36:20 PDT 2006
t2: Access Denied

Mon Sep  4 21:36:32 PDT 2006
t3: Access Denied

Mon Sep  4 21:36:43 PDT 2006
t4: Access Denied

Mon Sep  4 21:36:54 PDT 2006
t5: Access Denied

Static, file execute test, scenario 2 of 7
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:38:00 2006
Revoke access on: Mon Sep  4 21:38:10 2006

Target:  /usr/bin/cal
Grant access on:  Mon Sep  4 21:38:00 2006
Revoke access on: Mon Sep  4 21:38:10 2006

Mon Sep  4 21:37:50 PDT 2006
t1: Access Denied

Mon Sep  4 21:38:02 PDT 2006
   September 2006
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
```

```
17 18 19 20 21 22 23
24 25 26 27 28 29 30

t2: Access Granted

Mon Sep  4 21:38:14 PDT 2006
t3: Access Denied

Static, file execute test, scenario 3 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:39:29 2006
Revoke access on: Mon Sep  4 21:39:39 2006

Target:  /usr/bin/cal
Grant access on:  Mon Sep  4 21:39:39 2006
Revoke access on: Mon Sep  4 21:39:49 2006

Mon Sep  4 21:39:20 PDT 2006
t1: Access Denied

Mon Sep  4 21:39:31 PDT 2006
t2: Access Denied

Mon Sep  4 21:39:43 PDT 2006
t3: Access Denied

Mon Sep  4 21:39:54 PDT 2006
t4: Access Denied

Mon Sep  4 21:40:05 PDT 2006
t5: Access Denied

Static, file execute test, scenario 4 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:40:45 2006
Revoke access on: Mon Sep  4 21:41:05 2006

Target:  /usr/bin/cal
Grant access on:  Mon Sep  4 21:40:55 2006
Revoke access on: Mon Sep  4 21:41:15 2006

Mon Sep  4 21:40:35 PDT 2006
t1: Access Denied

Mon Sep  4 21:40:46 PDT 2006
t2: Access Denied

Mon Sep  4 21:40:58 PDT 2006
   September 2006
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
```

```
17 18 19 20 21 22 23
24 25 26 27 28 29 30

t3: Access Granted

Mon Sep  4 21:41:09 PDT 2006
t4: Access Denied

Mon Sep  4 21:41:21 PDT 2006
t5: Access Denied

done static exec file test, scenario 4 of 7.

Static, file write test, scenario 5 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 21:43:13 2006
Revoke access on: Mon Sep  4 21:43:23 2006

Target:  /usr/bin/cal
Grant access on:  Mon Sep  4 21:43:03 2006
Revoke access on: Mon Sep  4 21:43:33 2006

Mon Sep  4 21:42:54 PDT 2006
t1: Access Denied

Mon Sep  4 21:43:06 PDT 2006
t2: Access Denied

Mon Sep  4 21:43:18 PDT 2006
   September 2006
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

t3: Access Granted

Mon Sep  4 21:43:29 PDT 2006
t4: Access Denied

Mon Sep  4 21:43:40 PDT 2006
t5: Access Denied

done static exec file test, scenario 5 of 7.

Static, file write test, scenario 6 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:21:50 2006
Revoke access on: Mon Sep  4 22:22:00 2006

Target:  /usr/bin/cal
```

```
Grant access on:  Mon Sep  4 22:21:50 2006
Revoke access on: Mon Sep  4 22:22:10 2006

Mon Sep  4 22:21:40 PDT 2006
t1: Access Denied

Mon Sep  4 22:21:51 PDT 2006
   September 2006
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

t2: Access Granted

Mon Sep  4 22:22:03 PDT 2006
t3: Access Denied

Mon Sep  4 22:22:14 PDT 2006
t4: Access Denied

done exec file test, scenario 6 of 7

Static, file execute test, scenario 7 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:22:59 2006
Revoke access on: Mon Sep  4 22:23:09 2006

Target:  /usr/bin/cal
Grant access on:  Mon Sep  4 22:22:49 2006
Revoke access on: Mon Sep  4 22:23:09 2006

Mon Sep  4 22:22:40 PDT 2006
t1: Access Denied

Mon Sep  4 22:22:51 PDT 2006
t2: Access Denied

Mon Sep  4 22:23:03 PDT 2006
   September 2006
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

t3: Access Granted

Mon Sep  4 22:23:16 PDT 2006
t4: Access Denied

done static exec file test, scenario 7 of 7.
```

s-exec-dir--results.txt*

```
Static, directory execute test, scenario 1 of 7
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 22:48:40 2006
Revoke access on: Mon Sep  4 22:48:50 2006


setup object time attributes...
Target:  /tmp
Grant access on:  Mon Sep  4 22:49:01 2006
Revoke access on: Mon Sep  4 22:49:11 2006

Mon Sep  4 22:48:33 PDT 2006
t1: Access Denied

Mon Sep  4 22:48:44 PDT 2006
t2: Access Denied

Mon Sep  4 22:48:55 PDT 2006
t3: Access Denied

Mon Sep  4 22:49:06 PDT 2006
t4: Access Denied

Mon Sep  4 22:49:18 PDT 2006
t5: Access Denied

Static, directory execute test, scenario 2 of 7
setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 23:23:08 2006
Revoke access on: Mon Sep  4 23:23:18 2006

Target:  /tmp
Grant access on:  Mon Sep  4 23:23:08 2006
Revoke access on: Mon Sep  4 23:23:18 2006

Mon Sep  4 23:23:01 PDT 2006
t1: Access Denied

Mon Sep  4 23:23:13 PDT 2006
t2: Access Granted

Mon Sep  4 23:23:24 PDT 2006
t3: Access Denied

Static, directory execute test, scenario 3 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 23:26:44 2006
Revoke access on: Mon Sep  4 23:26:54 2006

Target:  /tmp
Grant access on:  Mon Sep  4 23:26:54 2006
```

```
Revoke access on: Mon Sep  4 23:27:04 2006

Mon Sep  4 23:26:35 PDT 2006
t1: Access Denied

Mon Sep  4 23:26:47 PDT 2006
t2: Access Denied

Mon Sep  4 23:26:59 PDT 2006
t3: Access Denied

Mon Sep  4 23:27:11 PDT 2006
t4: Access Denied

Mon Sep  4 23:27:23 PDT 2006
t5: Access Denied

Static, directory execute test, scenario 4 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 23:29:13 2006
Revoke access on: Mon Sep  4 23:29:33 2006

Target:  /tmp
Grant access on:  Mon Sep  4 23:29:24 2006
Revoke access on: Mon Sep  4 23:29:44 2006

Mon Sep  4 23:29:04 PDT 2006
t1: Access Denied

Mon Sep  4 23:29:16 PDT 2006
t2: Access Denied

Mon Sep  4 23:29:29 PDT 2006
t3: Access Granted

Mon Sep  4 23:29:41 PDT 2006
t4: Access Denied

Mon Sep  4 23:29:53 PDT 2006
t5: Access Denied

Static, directory execute test, scenario 5 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 23:30:34 2006
Revoke access on: Mon Sep  4 23:30:44 2006

Target:  /tmp
Grant access on:  Mon Sep  4 23:30:25 2006
Revoke access on: Mon Sep  4 23:30:55 2006

Mon Sep  4 23:30:15 PDT 2006
t1: Access Denied
```

```
Mon Sep  4 23:30:26 PDT 2006
t2: Access Denied

Mon Sep  4 23:30:38 PDT 2006
t3: Access Granted

Mon Sep  4 23:30:49 PDT 2006
t4: Access Denied

Mon Sep  4 23:31:00 PDT 2006
t5: Access Denied

Static, directory execute test, scenario 6 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 23:32:17 2006
Revoke access on: Mon Sep  4 23:32:27 2006

Target:  /tmp
Grant access on:  Mon Sep  4 23:32:17 2006
Revoke access on: Mon Sep  4 23:32:37 2006

Mon Sep  4 23:32:07 PDT 2006
t1: Access Denied

Mon Sep  4 23:32:19 PDT 2006
t2: Access Granted

Mon Sep  4 23:32:31 PDT 2006
t3: Access Denied

Mon Sep  4 23:32:43 PDT 2006
t4: Access Denied

Static, directory execute test, scenario 7 of 7

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Mon Sep  4 23:37:53 2006
Revoke access on: Mon Sep  4 23:38:03 2006

Target:  /tmp
Grant access on:  Mon Sep  4 23:37:43 2006
Revoke access on: Mon Sep  4 23:38:03 2006

Mon Sep  4 23:37:33 PDT 2006
t1: Access Denied

Mon Sep  4 23:37:45 PDT 2006
t2: Access Denied

Mon Sep  4 23:37:57 PDT 2006
t3: Access Granted

Mon Sep  4 23:38:08 PDT 2006
t4: Access Denied
```

s-read-file-6-swap-results.txt

```
Static, read file test, scenario 6 of 7(subject and objects time
swapped)

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 00:04:25 2006
Revoke access on: Thu Sep  7 00:04:45 2006

Target:  /tmp/message.txt
Grant access on:  Thu Sep  7 00:04:25 2006
Revoke access on: Thu Sep  7 00:04:35 2006

Thu Sep  7 00:04:16 PDT 2006
t1: Access Denied

Thu Sep  7 00:04:27 PDT 2006
this message will self destruct in 10 seconds...
t2: Access Granted

Thu Sep  7 00:04:37 PDT 2006
t3: Access Denied

Thu Sep  7 00:04:47 PDT 2006
t4: Access Denied

done read file test, scenario 6 of 7 ...
```

s-write-dir-4-swap-results.txt

```
Static, directory write test, scenario 4of7(subject and object time
swapped)

setup subject and object time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 00:08:46 2006
Revoke access on: Thu Sep  7 00:09:06 2006

/tmp does not currently have accessible TIFPS attributes
Target:  /tmp
Grant access on:  Thu Sep  7 00:08:37 2006
Revoke access on: Thu Sep  7 00:08:57 2006

Thu Sep  7 00:08:27 PDT 2006
t1: Access Denied

Thu Sep  7 00:08:38 PDT 2006
t2: Access Denied

Thu Sep  7 00:08:48 PDT 2006
t3: Access Granted
```

```
Thu Sep  7 00:08:59 PDT 2006
t4: Access Denied

Thu Sep  7 00:09:09 PDT 2006
t5: Access Denied

contents of /tmp directory
00:00:27-3.txt
00:08:48-3.txt
06:26:03-2.txt
17:28:34-1.txt
17:29:15-2.txt
17:32:57-5.txt
17:33:46-3.txt
17:36:24-2.txt
17:37:01-1.txt
22:38:20-2.txt
22:42:21-3.txt
22:44:06-3.txt
22:45:40-2.txt
22:47:47-3.txt
23:56:00-3.txt
23:59:10-3.txt
dest.txt
longfile.txt
mapping-kchiang
mapping-root
message.txt
source.txt
s-read-file-2-results.txt
write-expired.txt
written.txt
```

Static tests- Inheritance in file/directory creation and file copy operations

s-copy-file-1a-results.txt

Trial 1:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:11:25 2006
Revoke access on: Thu Sep 21 09:11:25 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:11:26 2006
Revoke access on: Fri Sep  8 09:11:26 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:11:26 2006
Revoke access on: Thu Sep  7 10:11:26 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
```

```
Thu Sep  7 09:11:48 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:11:26 2006
Revoke access on: Thu Sep  7 10:11:26 2006
```

Trial 2:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:11:54 2006
Revoke access on: Thu Sep 21 09:11:54 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:11:54 2006
Revoke access on: Fri Sep  8 09:11:54 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:11:55 2006
Revoke access on: Thu Sep  7 10:11:55 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:11:57 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:11:55 2006
Revoke access on: Thu Sep  7 10:11:55 2006
```

Trial 3:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:00 2006
Revoke access on: Thu Sep 21 09:12:00 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:00 2006
Revoke access on: Fri Sep  8 09:12:00 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:01 2006
Revoke access on: Thu Sep  7 10:12:01 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:03 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:01 2006
Revoke access on: Thu Sep  7 10:12:01 2006
```

Trial 4:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:05 2006
Revoke access on: Thu Sep 21 09:12:05 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:06 2006
Revoke access on: Fri Sep  8 09:12:06 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:06 2006
Revoke access on: Thu Sep  7 10:12:06 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:09 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:06 2006
Revoke access on: Thu Sep  7 10:12:06 2006
```

Trial 5:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:12 2006
Revoke access on: Thu Sep 21 09:12:12 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:12 2006
Revoke access on: Fri Sep  8 09:12:12 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:12 2006
Revoke access on: Thu Sep  7 10:12:12 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:14 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:12 2006
Revoke access on: Thu Sep  7 10:12:12 2006
```

Trial 6:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
```

```
Grant access on:  Thu Aug 24 09:12:16 2006
Revoke access on: Thu Sep 21 09:12:16 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:16 2006
Revoke access on: Fri Sep  8 09:12:16 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:17 2006
Revoke access on: Thu Sep  7 10:12:17 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:18 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:17 2006
Revoke access on: Thu Sep  7 10:12:17 2006
```

Trial 7:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:20 2006
Revoke access on: Thu Sep 21 09:12:20 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:21 2006
Revoke access on: Fri Sep  8 09:12:21 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:21 2006
Revoke access on: Thu Sep  7 10:12:21 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:22 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:21 2006
Revoke access on: Thu Sep  7 10:12:21 2006
```

Trial 8:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:24 2006
Revoke access on: Thu Sep 21 09:12:24 2006

setup source object time attributes...
Target:  source.txt
```

```
Grant access on:  Wed Sep  6 09:12:25 2006
Revoke access on: Fri Sep  8 09:12:25 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:25 2006
Revoke access on: Thu Sep  7 10:12:25 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:26 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:25 2006
Revoke access on: Thu Sep  7 10:12:25 2006
```

Trial 9:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:28 2006
Revoke access on: Thu Sep 21 09:12:28 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:29 2006
Revoke access on: Fri Sep  8 09:12:29 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:29 2006
Revoke access on: Thu Sep  7 10:12:29 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:30 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:29 2006
Revoke access on: Thu Sep  7 10:12:29 2006
```

Trial 10:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:12:32 2006
Revoke access on: Thu Sep 21 09:12:32 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:12:32 2006
Revoke access on: Fri Sep  8 09:12:32 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
```

```
Grant access on:  Thu Sep  7 08:12:33 2006
Revoke access on: Thu Sep  7 10:12:33 2006

Static, copy file test (using cp), scenario 1 of 3 - smallest dest
object

current time is:
Thu Sep  7 09:12:34 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:12:33 2006
Revoke access on: Thu Sep  7 10:12:33 2006
```

s-copy-file-1b-results.txt

Trial 1:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:16:43 2006
Revoke access on: Thu Sep 21 09:16:43 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:16:43 2006
Revoke access on: Fri Sep  8 09:16:43 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:16:43 2006
Revoke access on: Thu Sep  7 10:16:43 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:00 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:16:43 2006
Revoke access on: Thu Sep  7 10:16:43 2006
```

Trial 2:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:04 2006
Revoke access on: Thu Sep 21 09:17:04 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:04 2006
Revoke access on: Fri Sep  8 09:17:04 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:04 2006
Revoke access on: Thu Sep  7 10:17:04 2006
```

```
Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:06 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:04 2006
Revoke access on: Thu Sep  7 10:17:04 2006
```

Trial 3:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:08 2006
Revoke access on: Thu Sep 21 09:17:08 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:09 2006
Revoke access on: Fri Sep  8 09:17:09 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:09 2006
Revoke access on: Thu Sep  7 10:17:09 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:10 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:09 2006
Revoke access on: Thu Sep  7 10:17:09 2006
```

Trial 4:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:14 2006
Revoke access on: Thu Sep 21 09:17:14 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:15 2006
Revoke access on: Fri Sep  8 09:17:15 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:15 2006
Revoke access on: Thu Sep  7 10:17:15 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
```

```
Thu Sep  7 09:17:17 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:15 2006
Revoke access on: Thu Sep  7 10:17:15 2006
```

Trial 5:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:19 2006
Revoke access on: Thu Sep 21 09:17:19 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:19 2006
Revoke access on: Fri Sep  8 09:17:19 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:20 2006
Revoke access on: Thu Sep  7 10:17:20 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:21 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:20 2006
Revoke access on: Thu Sep  7 10:17:20 2006
```

Trial 6:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:23 2006
Revoke access on: Thu Sep 21 09:17:23 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:23 2006
Revoke access on: Fri Sep  8 09:17:23 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:24 2006
Revoke access on: Thu Sep  7 10:17:24 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:25 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:24 2006
Revoke access on: Thu Sep  7 10:17:24 2006
```

Trial 7:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:28 2006
Revoke access on: Thu Sep 21 09:17:28 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:28 2006
Revoke access on: Fri Sep  8 09:17:28 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:28 2006
Revoke access on: Thu Sep  7 10:17:28 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:30 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:28 2006
Revoke access on: Thu Sep  7 10:17:28 2006
```

Trial 8:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:32 2006
Revoke access on: Thu Sep 21 09:17:32 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:32 2006
Revoke access on: Fri Sep  8 09:17:32 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:33 2006
Revoke access on: Thu Sep  7 10:17:33 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:34 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:33 2006
Revoke access on: Thu Sep  7 10:17:33 2006
```

Trial 9:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
```

```
Grant access on:  Thu Aug 24 09:17:36 2006
Revoke access on: Thu Sep 21 09:17:36 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:36 2006
Revoke access on: Fri Sep  8 09:17:36 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:36 2006
Revoke access on: Thu Sep  7 10:17:36 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:38 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:36 2006
Revoke access on: Thu Sep  7 10:17:36 2006
```

Trial 10:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:17:40 2006
Revoke access on: Thu Sep 21 09:17:40 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:17:40 2006
Revoke access on: Fri Sep  8 09:17:40 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:40 2006
Revoke access on: Thu Sep  7 10:17:40 2006

Static, copy file test (using redirection), scenario 1 of 3 - smallest
dest object

current time is:
Thu Sep  7 09:17:42 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:17:40 2006
Revoke access on: Thu Sep  7 10:17:40 2006
```

s-copy-file-1c-results.txt

Trial 1:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:18:59 2006
Revoke access on: Thu Sep 21 09:18:59 2006
```

```
setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:18:59 2006
Revoke access on: Fri Sep  8 09:18:59 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:00 2006
Revoke access on: Thu Sep  7 10:19:00 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:19:19 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:00 2006
Revoke access on: Thu Sep  7 10:19:00 2006
```

Trial 2:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:19:24 2006
Revoke access on: Thu Sep 21 09:19:24 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:19:24 2006
Revoke access on: Fri Sep  8 09:19:24 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:24 2006
Revoke access on: Thu Sep  7 10:19:24 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:19:27 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:24 2006
Revoke access on: Thu Sep  7 10:19:24 2006
```

Trial 3:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:19:29 2006
Revoke access on: Thu Sep 21 09:19:29 2006

setup source object time attributes...
```

```
Target:  source.txt
Grant access on:  Wed Sep  6 09:19:30 2006
Revoke access on: Fri Sep  8 09:19:30 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:30 2006
Revoke access on: Thu Sep  7 10:19:30 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:19:31 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:30 2006
Revoke access on: Thu Sep  7 10:19:30 2006
```

Trial 4:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:19:34 2006
Revoke access on: Thu Sep 21 09:19:34 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:19:34 2006
Revoke access on: Fri Sep  8 09:19:34 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:34 2006
Revoke access on: Thu Sep  7 10:19:34 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:19:36 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:34 2006
Revoke access on: Thu Sep  7 10:19:34 2006
```

Trial 5:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:19:41 2006
Revoke access on: Thu Sep 21 09:19:41 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:19:41 2006
Revoke access on: Fri Sep  8 09:19:41 2006
```

```
setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:41 2006
Revoke access on: Thu Sep  7 10:19:41 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:19:43 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:41 2006
Revoke access on: Thu Sep  7 10:19:41 2006
```

Trial 6:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:19:46 2006
Revoke access on: Thu Sep 21 09:19:46 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:19:46 2006
Revoke access on: Fri Sep  8 09:19:46 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:47 2006
Revoke access on: Thu Sep  7 10:19:47 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:19:50 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:19:47 2006
Revoke access on: Thu Sep  7 10:19:47 2006
```

Trial 7:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:20:07 2006
Revoke access on: Thu Sep 21 09:20:07 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:20:07 2006
Revoke access on: Fri Sep  8 09:20:07 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
```

```
Grant access on:  Thu Sep  7 08:20:08 2006
Revoke access on: Thu Sep  7 10:20:08 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:20:09 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:08 2006
Revoke access on: Thu Sep  7 10:20:08 2006
```

Trial 8:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:20:12 2006
Revoke access on: Thu Sep 21 09:20:12 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:20:12 2006
Revoke access on: Fri Sep  8 09:20:12 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:13 2006
Revoke access on: Thu Sep  7 10:20:13 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:20:14 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:13 2006
Revoke access on: Thu Sep  7 10:20:13 2006
```

Trial 9:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:20:16 2006
Revoke access on: Thu Sep 21 09:20:16 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:20:16 2006
Revoke access on: Fri Sep  8 09:20:16 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:17 2006
Revoke access on: Thu Sep  7 10:20:17 2006
```

```
Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:20:19 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:17 2006
Revoke access on: Thu Sep  7 10:20:17 2006

Trial 10:
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:20:22 2006
Revoke access on: Thu Sep 21 09:20:22 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:20:22 2006
Revoke access on: Fri Sep  8 09:20:22 2006

setup destination object time attributes...(smallest)
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:22 2006
Revoke access on: Thu Sep  7 10:20:22 2006

Static, copy file test (using pipes), scenario 1 of 3 - smallest dest
obj

current time is:
Thu Sep  7 09:20:24 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:20:22 2006
Revoke access on: Thu Sep  7 10:20:22 2006
```

s-copy-file-2a-results.txt

```
Trial 1:
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:28:17 2006
Revoke access on: Thu Sep 21 09:28:17 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:28:17 2006
Revoke access on: Thu Sep  7 10:28:17 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:28:17 2006
Revoke access on: Fri Sep  8 09:28:17 2006
```

```
Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:28:30 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:28:17 2006
Revoke access on: Thu Sep  7 10:28:17 2006
```

Trial 2:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:28:34 2006
Revoke access on: Thu Sep 21 09:28:34 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:28:34 2006
Revoke access on: Thu Sep  7 10:28:34 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:28:34 2006
Revoke access on: Fri Sep  8 09:28:34 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:28:36 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:28:34 2006
Revoke access on: Thu Sep  7 10:28:34 2006
```

Trial 3:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:28:38 2006
Revoke access on: Thu Sep 21 09:28:38 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:28:39 2006
Revoke access on: Thu Sep  7 10:28:39 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:28:39 2006
Revoke access on: Fri Sep  8 09:28:39 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:28:40 PDT 2006
```

```
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:28:39 2006
Revoke access on: Thu Sep  7 10:28:39 2006
```

Trial 4:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:28:42 2006
Revoke access on: Thu Sep 21 09:28:42 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:28:42 2006
Revoke access on: Thu Sep  7 10:28:42 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:28:43 2006
Revoke access on: Fri Sep  8 09:28:43 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:28:44 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:28:42 2006
Revoke access on: Thu Sep  7 10:28:42 2006
```

Trial 5:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:28:45 2006
Revoke access on: Thu Sep 21 09:28:45 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:28:46 2006
Revoke access on: Thu Sep  7 10:28:46 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:28:46 2006
Revoke access on: Fri Sep  8 09:28:46 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:28:57 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:28:46 2006
Revoke access on: Thu Sep  7 10:28:46 2006
```

Trial 6:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:28:58 2006
Revoke access on: Thu Sep 21 09:28:58 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:28:59 2006
Revoke access on: Thu Sep  7 10:28:59 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:28:59 2006
Revoke access on: Fri Sep  8 09:28:59 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:29:00 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:28:59 2006
Revoke access on: Thu Sep  7 10:28:59 2006
```

Trial 7:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:29:02 2006
Revoke access on: Thu Sep 21 09:29:02 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:29:02 2006
Revoke access on: Thu Sep  7 10:29:02 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:29:02 2006
Revoke access on: Fri Sep  8 09:29:02 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:29:03 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:29:02 2006
Revoke access on: Thu Sep  7 10:29:02 2006
```

Trial 8:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:29:05 2006
Revoke access on: Thu Sep 21 09:29:05 2006
```

```
setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:29:05 2006
Revoke access on: Thu Sep  7 10:29:05 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:29:05 2006
Revoke access on: Fri Sep  8 09:29:05 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:29:06 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:29:05 2006
Revoke access on: Thu Sep  7 10:29:05 2006
```

Trial 9:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:29:08 2006
Revoke access on: Thu Sep 21 09:29:08 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:29:08 2006
Revoke access on: Thu Sep  7 10:29:08 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:29:08 2006
Revoke access on: Fri Sep  8 09:29:08 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:29:09 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:29:08 2006
Revoke access on: Thu Sep  7 10:29:08 2006
```

Trial 10:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:29:11 2006
Revoke access on: Thu Sep 21 09:29:11 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:29:11 2006
Revoke access on: Thu Sep  7 10:29:11 2006
```

```
setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:29:11 2006
Revoke access on: Fri Sep  8 09:29:11 2006

Static, copy file test (using cp), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 09:29:13 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:29:11 2006
Revoke access on: Thu Sep  7 10:29:11 2006
```

s-copy-file-2b-results.txt

Trial 1:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:04 2006
Revoke access on: Thu Sep 21 09:26:04 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:04 2006
Revoke access on: Thu Sep  7 10:26:04 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:04 2006
Revoke access on: Fri Sep  8 09:26:04 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:14 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:04 2006
Revoke access on: Thu Sep  7 10:26:04 2006
```

Trial 2:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:16 2006
Revoke access on: Thu Sep 21 09:26:16 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:16 2006
Revoke access on: Thu Sep  7 10:26:16 2006

setup destination object time attributes...
```

```
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:16 2006
Revoke access on: Fri Sep  8 09:26:16 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:18 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:16 2006
Revoke access on: Thu Sep  7 10:26:16 2006
```

Trial 3:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:20 2006
Revoke access on: Thu Sep 21 09:26:20 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:20 2006
Revoke access on: Thu Sep  7 10:26:20 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:21 2006
Revoke access on: Fri Sep  8 09:26:21 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:22 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:20 2006
Revoke access on: Thu Sep  7 10:26:20 2006
```

Trial 4:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:24 2006
Revoke access on: Thu Sep 21 09:26:24 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:24 2006
Revoke access on: Thu Sep  7 10:26:24 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:25 2006
Revoke access on: Fri Sep  8 09:26:25 2006
```

```
Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:26 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:24 2006
Revoke access on: Thu Sep  7 10:26:24 2006
```

Trial 5:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:27 2006
Revoke access on: Thu Sep 21 09:26:27 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:27 2006
Revoke access on: Thu Sep  7 10:26:27 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:28 2006
Revoke access on: Fri Sep  8 09:26:28 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:29 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:27 2006
Revoke access on: Thu Sep  7 10:26:27 2006
```

Trial 6:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:30 2006
Revoke access on: Thu Sep 21 09:26:30 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:31 2006
Revoke access on: Thu Sep  7 10:26:31 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:31 2006
Revoke access on: Fri Sep  8 09:26:31 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:32 PDT 2006
```

```
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:31 2006
Revoke access on: Thu Sep  7 10:26:31 2006
```

Trial 7:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:35 2006
Revoke access on: Thu Sep 21 09:26:35 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:35 2006
Revoke access on: Thu Sep  7 10:26:35 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:35 2006
Revoke access on: Fri Sep  8 09:26:35 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:37 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:35 2006
Revoke access on: Thu Sep  7 10:26:35 2006
```

Trial 8:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:39 2006
Revoke access on: Thu Sep 21 09:26:39 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:40 2006
Revoke access on: Thu Sep  7 10:26:40 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:40 2006
Revoke access on: Fri Sep  8 09:26:40 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:42 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:40 2006
Revoke access on: Thu Sep  7 10:26:40 2006
```

Trial 9:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:47 2006
Revoke access on: Thu Sep 21 09:26:47 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:47 2006
Revoke access on: Thu Sep  7 10:26:47 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:47 2006
Revoke access on: Fri Sep  8 09:26:47 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:49 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:47 2006
Revoke access on: Thu Sep  7 10:26:47 2006
```

Trial 10:
```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 09:26:51 2006
Revoke access on: Thu Sep 21 09:26:51 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 08:26:51 2006
Revoke access on: Thu Sep  7 10:26:51 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:26:51 2006
Revoke access on: Fri Sep  8 09:26:51 2006

Static, copy file test (using redirection), scenario 2 of 3 - smallest
src object

current time is:
Thu Sep  7 09:26:53 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:26:51 2006
Revoke access on: Thu Sep  7 10:26:51 2006
```

s-copy-file-2c-results.txt

Trial 1:
```
setup subject time attributes...
```

```
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 13:15:46 2006
Revoke access on: Thu Sep 21 13:15:46 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 12:15:46 2006
Revoke access on: Thu Sep  7 14:15:46 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:15:47 2006
Revoke access on: Fri Sep  8 13:15:47 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 13:16:17 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 12:15:46 2006
Revoke access on: Thu Sep  7 14:15:46 2006
```

Trial 2:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 13:55:44 2006
Revoke access on: Thu Sep 21 13:55:44 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 12:55:45 2006
Revoke access on: Thu Sep  7 14:55:45 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:55:45 2006
Revoke access on: Fri Sep  8 13:55:45 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 13:55:54 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 12:55:45 2006
Revoke access on: Thu Sep  7 14:55:45 2006
```

Trial 3:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 13:55:58 2006
Revoke access on: Thu Sep 21 13:55:58 2006
```

```
setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 12:55:59 2006
Revoke access on: Thu Sep  7 14:55:59 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:55:59 2006
Revoke access on: Fri Sep  8 13:55:59 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 13:56:03 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 12:55:59 2006
Revoke access on: Thu Sep  7 14:55:59 2006
```

Trial 4:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 13:56:13 2006
Revoke access on: Thu Sep 21 13:56:13 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 12:56:13 2006
Revoke access on: Thu Sep  7 14:56:13 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:56:13 2006
Revoke access on: Fri Sep  8 13:56:13 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 13:56:19 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 12:56:13 2006
Revoke access on: Thu Sep  7 14:56:13 2006
```

Trial 5:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 13:56:33 2006
Revoke access on: Thu Sep 21 13:56:33 2006

setup source object time attributes... (smallest)
Target:  source.txt
```

```
Grant access on:  Thu Sep  7 12:56:33 2006
Revoke access on: Thu Sep  7 14:56:33 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:56:33 2006
Revoke access on: Fri Sep  8 13:56:33 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 13:56:37 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:56:33 2006
Revoke access on: Fri Sep  8 13:56:33 2006

Trial 6:
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 13:59:57 2006
Revoke access on: Thu Sep 21 13:59:57 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 12:59:58 2006
Revoke access on: Thu Sep  7 14:59:58 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 13:59:58 2006
Revoke access on: Fri Sep  8 13:59:58 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 14:00:03 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 12:59:58 2006
Revoke access on: Thu Sep  7 14:59:58 2006

Trial 7:
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:09 2006
Revoke access on: Thu Sep 21 14:00:09 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 13:00:09 2006
Revoke access on: Thu Sep  7 15:00:09 2006
```

```
setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 14:00:09 2006
Revoke access on: Fri Sep  8 14:00:09 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 14:00:14 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 13:00:09 2006
Revoke access on: Thu Sep  7 15:00:09 2006
```

Trial 8:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:18 2006
Revoke access on: Thu Sep 21 14:00:18 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 13:00:19 2006
Revoke access on: Thu Sep  7 15:00:19 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 14:00:19 2006
Revoke access on: Fri Sep  8 14:00:19 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 14:00:23 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 13:00:19 2006
Revoke access on: Thu Sep  7 15:00:19 2006
```

Trial 9:

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:36 2006
Revoke access on: Thu Sep 21 14:00:36 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 13:00:36 2006
Revoke access on: Thu Sep  7 15:00:36 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 14:00:36 2006
```

```
Revoke access on: Fri Sep  8 14:00:36 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 14:00:40 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 13:00:36 2006
Revoke access on: Thu Sep  7 15:00:36 2006

Trial 10:
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:45 2006
Revoke access on: Thu Sep 21 14:00:45 2006

setup source object time attributes... (smallest)
Target:  source.txt
Grant access on:  Thu Sep  7 13:00:46 2006
Revoke access on: Thu Sep  7 15:00:46 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 14:00:46 2006
Revoke access on: Fri Sep  8 14:00:46 2006

Static, copy file test (using pipes), scenario 2 of 3 - smallest src
object

current time is:
Thu Sep  7 14:00:49 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 13:00:46 2006
Revoke access on: Thu Sep  7 15:00:46 2006
```

s-copy-file-3a-results.txt

```
Trial 1:
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:33:52 2006
Revoke access on: Thu Sep  7 10:33:52 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:33:52 2006
Revoke access on: Fri Sep  8 09:33:52 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:33:52 2006
```

```
Revoke access on: Fri Sep  8 09:33:52 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:34:08 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:33:52 2006
Revoke access on: Thu Sep  7 10:33:52 2006
```

Trial 2:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:34:37 2006
Revoke access on: Thu Sep  7 10:34:37 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:34:37 2006
Revoke access on: Fri Sep  8 09:34:37 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:34:37 2006
Revoke access on: Fri Sep  8 09:34:37 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:34:47 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:34:37 2006
Revoke access on: Thu Sep  7 10:34:37 2006
```

Trial 3:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:34:52 2006
Revoke access on: Thu Sep  7 10:34:52 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:34:52 2006
Revoke access on: Fri Sep  8 09:34:52 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:34:52 2006
Revoke access on: Fri Sep  8 09:34:52 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:35:00 PDT 2006
The resulting time attribute for the destination file is ...
```

```
Target:  dest.txt
Grant access on:  Thu Sep  7 08:34:52 2006
Revoke access on: Thu Sep  7 10:34:52 2006
```

Trial 4:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:35:04 2006
Revoke access on: Thu Sep  7 10:35:04 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:35:05 2006
Revoke access on: Fri Sep  8 09:35:05 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:35:05 2006
Revoke access on: Fri Sep  8 09:35:05 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:35:10 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:35:04 2006
Revoke access on: Thu Sep  7 10:35:04 2006
```

Trial 5:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:42:19 2006
Revoke access on: Thu Sep  7 10:42:19 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:42:20 2006
Revoke access on: Fri Sep  8 09:42:20 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:42:20 2006
Revoke access on: Fri Sep  8 09:42:20 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:42:30 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:42:19 2006
Revoke access on: Thu Sep  7 10:42:19 2006
```

Trial 6:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
```

```
Grant access on:  Thu Sep  7 08:42:40 2006
Revoke access on: Thu Sep  7 10:42:40 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:42:40 2006
Revoke access on: Fri Sep  8 09:42:40 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:42:41 2006
Revoke access on: Fri Sep  8 09:42:41 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:42:45 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:42:40 2006
Revoke access on: Thu Sep  7 10:42:40 2006
```

Trial 7:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:42:49 2006
Revoke access on: Thu Sep  7 10:42:49 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:42:49 2006
Revoke access on: Fri Sep  8 09:42:49 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:42:50 2006
Revoke access on: Fri Sep  8 09:42:50 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:42:55 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:42:49 2006
Revoke access on: Thu Sep  7 10:42:49 2006
```

Trial 8:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:43:00 2006
Revoke access on: Thu Sep  7 10:43:00 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:43:00 2006
Revoke access on: Fri Sep  8 09:43:00 2006
```

```
setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:43:01 2006
Revoke access on: Fri Sep  8 09:43:01 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:43:08 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:43:00 2006
Revoke access on: Thu Sep  7 10:43:00 2006
```

Trial 9:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:43:17 2006
Revoke access on: Thu Sep  7 10:43:17 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:43:18 2006
Revoke access on: Fri Sep  8 09:43:18 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:43:18 2006
Revoke access on: Fri Sep  8 09:43:18 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject

current time is:
Thu Sep  7 09:43:23 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:43:17 2006
Revoke access on: Thu Sep  7 10:43:17 2006
```

Trial 10:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:43:28 2006
Revoke access on: Thu Sep  7 10:43:28 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:43:28 2006
Revoke access on: Fri Sep  8 09:43:28 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:43:29 2006
Revoke access on: Fri Sep  8 09:43:29 2006

Static, copy file test (using cp), scenario 3 of 3 - smallest subject
```

```
current time is:
Thu Sep  7 09:43:35 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:43:28 2006
Revoke access on: Thu Sep  7 10:43:28 2006
```

s-copy-file-3b-results.txt

Trial 1:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:44:52 2006
Revoke access on: Thu Sep  7 10:44:52 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:44:53 2006
Revoke access on: Fri Sep  8 09:44:53 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:44:53 2006
Revoke access on: Fri Sep  8 09:44:53 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:45:06 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:44:52 2006
Revoke access on: Thu Sep  7 10:44:52 2006
```

Trial 2:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:45:13 2006
Revoke access on: Thu Sep  7 10:45:13 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:45:13 2006
Revoke access on: Fri Sep  8 09:45:13 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:45:14 2006
Revoke access on: Fri Sep  8 09:45:14 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
```

```
Thu Sep  7 09:45:19 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:45:13 2006
Revoke access on: Thu Sep  7 10:45:13 2006
```

Trial 3:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:45:24 2006
Revoke access on: Thu Sep  7 10:45:24 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:45:25 2006
Revoke access on: Fri Sep  8 09:45:25 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:45:25 2006
Revoke access on: Fri Sep  8 09:45:25 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:45:31 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:45:24 2006
Revoke access on: Thu Sep  7 10:45:24 2006
```

Trial 4:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:45:38 2006
Revoke access on: Thu Sep  7 10:45:38 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:45:39 2006
Revoke access on: Fri Sep  8 09:45:39 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:45:39 2006
Revoke access on: Fri Sep  8 09:45:39 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:45:43 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:45:38 2006
Revoke access on: Thu Sep  7 10:45:38 2006
```

Trial 5:
```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:45:52 2006
Revoke access on: Thu Sep  7 10:45:52 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:45:52 2006
Revoke access on: Fri Sep  8 09:45:52 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:45:53 2006
Revoke access on: Fri Sep  8 09:45:53 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:45:57 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:45:52 2006
Revoke access on: Thu Sep  7 10:45:52 2006
```

Trial 6:
```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:46:01 2006
Revoke access on: Thu Sep  7 10:46:01 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:46:02 2006
Revoke access on: Fri Sep  8 09:46:02 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:46:02 2006
Revoke access on: Fri Sep  8 09:46:02 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:46:06 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:46:01 2006
Revoke access on: Thu Sep  7 10:46:01 2006
```

Trial 7:
```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:46:10 2006
```

```
Revoke access on: Thu Sep  7 10:46:10 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:46:11 2006
Revoke access on: Fri Sep  8 09:46:11 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:46:11 2006
Revoke access on: Fri Sep  8 09:46:11 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:46:18 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:46:10 2006
Revoke access on: Thu Sep  7 10:46:10 2006

Trial                                                                 8:
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:46:26 2006
Revoke access on: Thu Sep  7 10:46:26 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:46:26 2006
Revoke access on: Fri Sep  8 09:46:26 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:46:26 2006
Revoke access on: Fri Sep  8 09:46:26 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:46:33 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:46:26 2006
Revoke access on: Thu Sep  7 10:46:26 2006

Trial 9:
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:46:41 2006
Revoke access on: Thu Sep  7 10:46:41 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:46:41 2006
```

```
Revoke access on: Fri Sep  8 09:46:41 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:46:42 2006
Revoke access on: Fri Sep  8 09:46:42 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:47:29 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:46:41 2006
Revoke access on: Thu Sep  7 10:46:41 2006
```

Trial 10:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:47:34 2006
Revoke access on: Thu Sep  7 10:47:34 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:47:35 2006
Revoke access on: Fri Sep  8 09:47:35 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:47:35 2006
Revoke access on: Fri Sep  8 09:47:35 2006

Static, copy file test (using redirection), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:47:39 PDT 2006
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:47:34 2006
Revoke access on: Thu Sep  7 10:47:34 2006
```

s-copy-file-3c-results.txt

Trial 1:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:49:17 2006
Revoke access on: Thu Sep  7 10:49:17 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:49:17 2006
Revoke access on: Fri Sep  8 09:49:17 2006
```

```
setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:49:17 2006
Revoke access on: Fri Sep  8 09:49:17 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:49:31 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:49:17 2006
Revoke access on: Thu Sep  7 10:49:17 2006
```

Trial 2:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:49:38 2006
Revoke access on: Thu Sep  7 10:49:38 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:49:39 2006
Revoke access on: Fri Sep  8 09:49:39 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:49:39 2006
Revoke access on: Fri Sep  8 09:49:39 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:49:44 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:49:38 2006
Revoke access on: Thu Sep  7 10:49:38 2006
```

Trial 3:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:49:49 2006
Revoke access on: Thu Sep  7 10:49:49 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:49:49 2006
Revoke access on: Fri Sep  8 09:49:49 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:49:50 2006
```

```
Revoke access on: Fri Sep  8 09:49:50 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:49:53 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:49:49 2006
Revoke access on: Thu Sep  7 10:49:49 2006
```

Trial 4:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:49:57 2006
Revoke access on: Thu Sep  7 10:49:57 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:49:58 2006
Revoke access on: Fri Sep  8 09:49:58 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:49:58 2006
Revoke access on: Fri Sep  8 09:49:58 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:50:02 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:49:57 2006
Revoke access on: Thu Sep  7 10:49:57 2006
```

Trial 5:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:50:05 2006
Revoke access on: Thu Sep  7 10:50:05 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:50:06 2006
Revoke access on: Fri Sep  8 09:50:06 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:50:06 2006
Revoke access on: Fri Sep  8 09:50:06 2006
```

```
Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:50:11 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:50:05 2006
Revoke access on: Thu Sep  7 10:50:05 2006
```

Trial 6:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:50:15 2006
Revoke access on: Thu Sep  7 10:50:15 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:50:16 2006
Revoke access on: Fri Sep  8 09:50:16 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:50:16 2006
Revoke access on: Fri Sep  8 09:50:16 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:50:22 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:50:15 2006
Revoke access on: Thu Sep  7 10:50:15 2006
```

Trial 7:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:50:34 2006
Revoke access on: Thu Sep  7 10:50:34 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:50:34 2006
Revoke access on: Fri Sep  8 09:50:34 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:50:34 2006
Revoke access on: Fri Sep  8 09:50:34 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject
```

```
current time is:
Thu Sep  7 09:50:38 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:50:34 2006
Revoke access on: Thu Sep  7 10:50:34 2006
```

Trial 8:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:50:43 2006
Revoke access on: Thu Sep  7 10:50:43 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:50:43 2006
Revoke access on: Fri Sep  8 09:50:43 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:50:43 2006
Revoke access on: Fri Sep  8 09:50:43 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:50:47 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:50:43 2006
Revoke access on: Thu Sep  7 10:50:43 2006
```

Trial 9:

```
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:50:50 2006
Revoke access on: Thu Sep  7 10:50:50 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:50:50 2006
Revoke access on: Fri Sep  8 09:50:50 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:50:50 2006
Revoke access on: Fri Sep  8 09:50:50 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:50:56 PDT 2006
This is the source file.
```

```
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:50:50 2006
Revoke access on: Thu Sep  7 10:50:50 2006

Trial 10:
setup subject time attributes...(smallest)
Target:  /home/jody/.bash_profile
Grant access on:  Thu Sep  7 08:50:59 2006
Revoke access on: Thu Sep  7 10:50:59 2006

setup source object time attributes...
Target:  source.txt
Grant access on:  Wed Sep  6 09:50:59 2006
Revoke access on: Fri Sep  8 09:50:59 2006

setup destination object time attributes...
Target:  dest.txt
Grant access on:  Wed Sep  6 09:51:00 2006
Revoke access on: Fri Sep  8 09:51:00 2006

Static, copy file test (using pipes), scenario 3 of 3 - smallest
subject

current time is:
Thu Sep  7 09:51:05 PDT 2006
This is the source file.
The resulting time attribute for the destination file is ...
Target:  dest.txt
Grant access on:  Thu Sep  7 08:50:59 2006
Revoke access on: Thu Sep  7 10:50:59 2006
```

s-create-file-results.txt

```
Static, create file test


setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Tue Aug 22 00:14:50 2006
Revoke access on: Tue Sep 19 00:14:50 2006


current time is:
Tue Sep  5 00:14:51 PDT 2006

creating a new file..........

The time attribute for the newly created file is ...
Target:  /home/jody/jodynew.txt
Grant access on:  Tue Aug 22 00:14:50 2006
Revoke access on: Tue Sep 19 00:14:50 2006
```

s-create-dir-results.txt

```
Static, create directory test
```

```
setup subject time attributes...
Target:  /home/jody/.bash_profile
Grant access on:  Tue Aug 22 00:15:47 2006
Revoke access on: Tue Sep 19 00:15:47 2006

current time is:
Tue Sep  5 00:15:47 PDT 2006

creating a new directory...........

The time attribute for the newly created directory is ...
Target:  /home/jody/jodyNewDirectory
Grant access on:  Tue Aug 22 00:15:47 2006
Revoke access on: Tue Sep 19 00:15:47 2006
```

Static tests – TIFPS behavior on time expiration during file write operations

```
[root@laptopthesisdev accesscontrol]# ./s-write-expire.sh 1
Static test: File expiration during write operation

setup subject and object time attributes...
file will expire in 1 second(s)
getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:45 2006
Revoke access on: Sun Sep 17 15:25:50 2006

getfattr: Removing leading '/' from absolute path names
Target:  /tmp/write-expired.txt
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Sun Sep 10 15:25:51 2006

write operation started:
Sun Sep 10 15:25:51 PDT 2006

User jody tries to append 5 million G's to /tmp/write-expired.txt file ..
ERROR opening file:  goodbye!

write operation ended:
Sun Sep 10 15:25:51 PDT 2006
Number of characters written to the file successfully:
0 /tmp/write-expired.txt

[root@laptopthesisdev accesscontrol]#
```

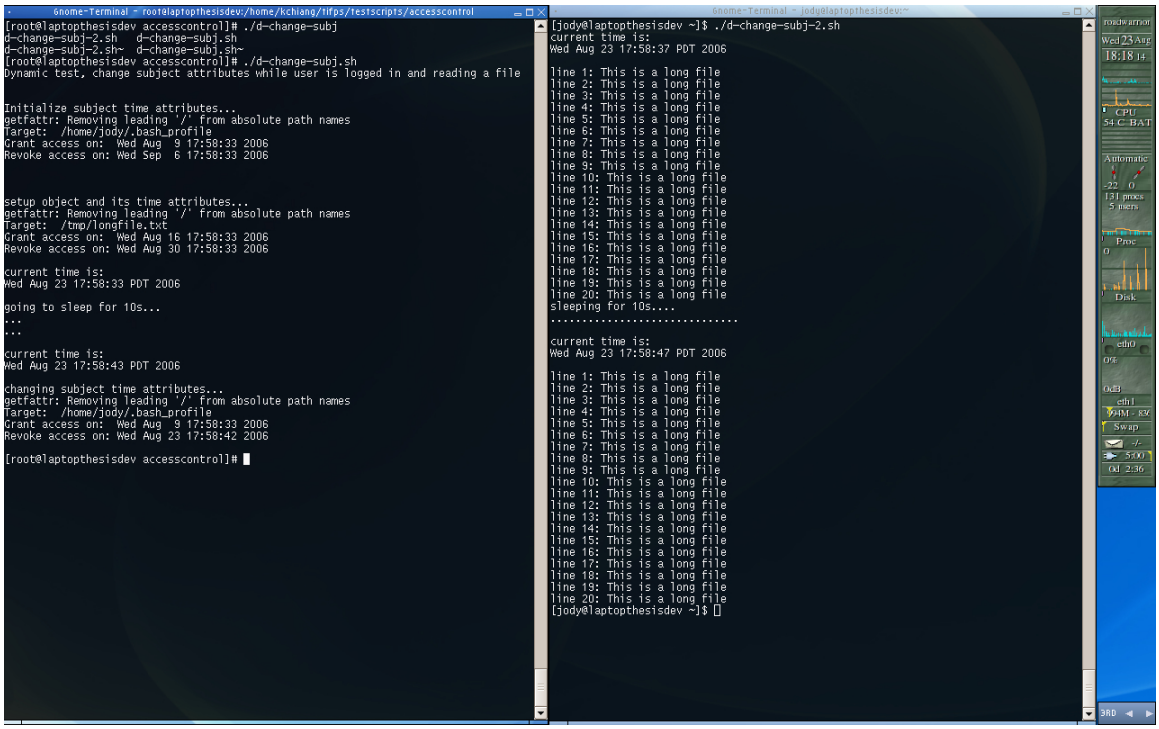
```
[root@laptopthesisdev accesscontrol]# ./s-write-expire.sh 2
Static test: File expiration during write operation

setup subject and object time attributes...
file will expire in 2 second(s)
getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:45 2006
Revoke access on: Sun Sep 17 15:26:10 2006

getfattr: Removing leading '/' from absolute path names
Target:  /tmp/write-expired.txt
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Sun Sep 10 15:26:12 2006

write operation started:
Sun Sep 10 15:26:11 PDT 2006

User jody tries to append 5 million G's to /tmp/write-expired.txt file ..
ERROR writing to file: ERR -1

write operation ended:
Sun Sep 10 15:26:12 PDT 2006
Number of characters written to the file successfully:
49152 /tmp/write-expired.txt

[root@laptopthesisdev accesscontrol]#
```

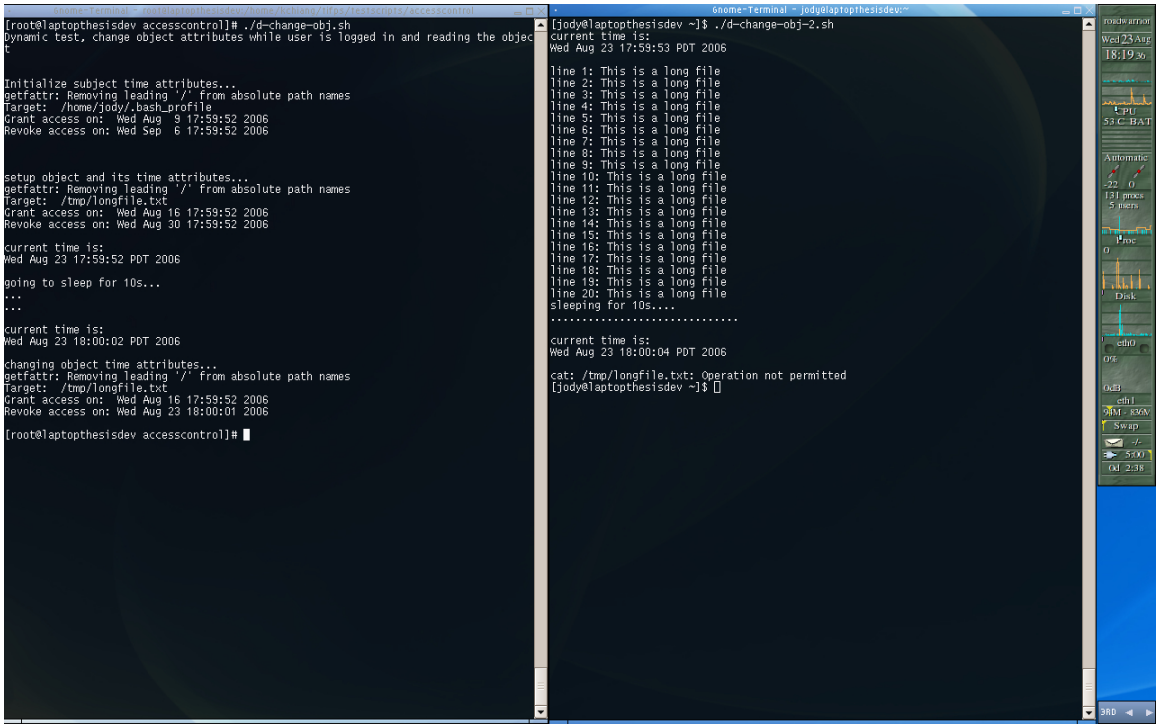
```
[root@laptopthesisdev accesscontrol]# ./s-write-expire.sh 3
Static test: File expiration during write operation

setup subject and object time attributes...
file will expire in 3 second(s)
getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:45 2006
Revoke access on: Sun Sep 17 15:26:31 2006

getfattr: Removing leading '/' from absolute path names
Target:  /tmp/write-expired.txt
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Sun Sep 10 15:26:35 2006

write operation started:
Sun Sep 10 15:26:32 PDT 2006

User jody tries to append 5 million G's to /tmp/write-expired.txt file ..
ERROR writing to file: ERR -1

write operation ended:
Sun Sep 10 15:26:35 PDT 2006
Number of characters written to the file successfully:
2002944 /tmp/write-expired.txt

[root@laptopthesisdev accesscontrol]#
```

```
[root@laptopthesisdev accesscontrol]# ./s-write-expire.sh 4
Static test: File expiration during write operation

setup subject and object time attributes...
file will expire in 4 second(s)
getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:45 2006
Revoke access on: Sun Sep 17 15:26:43 2006

getfattr: Removing leading '/' from absolute path names
Target:  /tmp/write-expired.txt
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Sun Sep 10 15:26:48 2006

write operation started:
Sun Sep 10 15:26:44 PDT 2006

User jody tries to append 5 million G's to /tmp/write-expired.txt file ..
ERROR writing to file: ERR -1

write operation ended:
Sun Sep 10 15:26:48 PDT 2006
Number of characters written to the file successfully:
3338240 /tmp/write-expired.txt

[root@laptopthesisdev accesscontrol]#
```

```
[root@laptopthesisdev accesscontrol]# ./s-write-expire.sh 5
Static test: File expiration during write operation

setup subject and object time attributes...
file will expire in 5 second(s)
getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/.bash_profile
Grant access on:  Thu Aug 24 14:00:45 2006
Revoke access on: Sun Sep 17 15:27:24 2006

getfattr: Removing leading '/' from absolute path names
Target:  /tmp/write-expired.txt
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Sun Sep 10 15:27:30 2006

write operation started:
Sun Sep 10 15:27:25 PDT 2006

User jody tries to append 5 million G's to /tmp/write-expired.txt file ..
File write successfully completed!

write operation ended:
Sun Sep 10 15:27:29 PDT 2006
Number of characters written to the file successfully:
5000000 /tmp/write-expired.txt

[root@laptopthesisdev accesscontrol]#
```

Dynamic tests – Dynamically changing subject and object attributes





D. PERFORMANCE TEST PROCEDURES

This section contains detailed test procedures for performance test evaluation of the TIFPS LSM.

1. The performance tests consist of six test scripts and twelve setup scripts. Prior to conducting the tests, ensure the following preconditions are met:

   a. Start two separate login sessions, one as ***root*** and the other as user ***sam.***

   b. In the ***root*** session, copy the scripts in the **testscripts/performance**/ directory to **/home/sam**/:

      # cp –r <path to testscripts directory>/performance /home/sam/

   c. Change both login sessions to the **/home/sam/performance/** directory.

   d. In the ***root*** session, change the permission of the **testscripts/performance/** directory:

      # chmod –R 777 /home/sam/performance

   e. Unload the TIFPS LSM from the kernel using the ***root*** session:

      # rmmod tifps

2. In the ***root*** login session, run the setup script:

   # ./test1setup.sh

3. In the ***sam*** login session, run the following command three times and record the resulting *sys* time for each trial (this will be the baseline performance for a kernel without TIFPS):

   $ time –p ./test1and2.sh

4. In the ***root*** login session, load the TIFPS LSM by running:

   # modprobe tifps

5. In the ***sam*** login session, run the same command as in step 3 three times and record the resulting *sys* time for each trial (this will be the performance for a kernel with TIFPS)

$ time –p ./test1and2.sh

6. Repeat steps 1 through 5 for subsequent tests 3, 5, 7, 9, and 11.

7. For tests 2, 4, 6, 10, and 12, repeat steps 1 through 5 with one change – run the **test*setup.sh** after every trial in steps 3 and 5.

8. For test 8, this test does not have a setup script, to run these test 8, edit the **test7and8.sh** script by un-commenting the last line in the script.

9. Test scripts for each performance test condition are summarized in Table D-7.

Table D-7. Summary of test scripts used for each performance evaluation condition

TestID	Test Scripts	Performance test variable descriptions
F1	test1setup.sh test1and2.sh	Read single file with TIFPS attributes 1000 times
F2	test2setup.sh test1and2.sh	Read single file without TIFPS attributes 1000 times
F3	test3setup.sh test3and4.sh	Read 1000 files with TIFPS attributes 1 time
F4	test4setup.sh test3and4.sh	Read 1000 files without TIFPS attributes 1 time
F5	test5setup.sh test5and6.sh	Write single file with TIFPS attributes 1000 times
F6	test6setup.sh test5and6.sh	Write single file without TIFPS attributes 1000 times
F7	test7setup.sh test7and8.sh	Write 1000 files with TIFPS attributes 1 time
F8	test7and8.sh	Write 1000 files without TIFPS attributes 1 time
F9	test9setup.sh test9and10.sh	Copy 1 file with TIFPS attributes 1000 times to another existing file with TIFPS attributes
F10	test10setup.sh test9and10.sh	Copy 1 file without TIFPS attributes 1000 times to another non existent file
F11	test11setup.sh test11and12.sh	Copy 1000 different files, each with TIFPS attributes to another set of 1000 files, with TIFPS attributes
F12	test12setup.sh test11and12.sh	Copy 1000 different files, without TIFPS attributes to a set of non existent files

E. PERFORMANCE TEST SCRIPTS

This section contains scripts that correspond to the tests described in Section D.

test1and2.sh

```
#!/bin/bash
# For test 1, run test1setup.sh once as root to create the file to be read and
# set the time attributes
# For test 2, run test2setup.sh for every trial as root before running
# test1and2.sh

i=1
while [ $i -lt 1000 ];do
    cat test1message.txt >/dev/null
    let i++
done
```

test3and4.sh

```
#!/bin/bash
# For test 3, run test3setup.sh once as root to create 1000 files
#  for this script
# For test 4, run test4setup.sh between every trial as root before running
# this script.
i=1
while [ $i -lt 1000 ]; do
    cat test3-file$i.txt >/dev/null
    let i++
done
```

test5and6.sh

```
#!/bin/bash
# For test 5, run the test5setup.sh script as root once before running this.
# For test 6, run the test6setup.sh script as root between every trial before
#  running this script.
i=1
while [ $i -lt 1000 ];do
    python -c “print 'G'*1000”> writefile.txt
    let i++
done
```

test7and8.sh

```
#!/bin/bash

# For test 7, run the test7setup.sh script first to setup the test.
# For test 8, uncomment the last line for test 8 before running this test.

i=1
while [ $i -lt 1000 ];do
    python -c “print 'G'*1000” > “test7-8-file$i.txt”
    let i++
done

#uncomment following for test 8
#rm -rf test7-8-file*  #test 8 only
```

test9and10.sh

```
#!/bin/bash
# For test 9, run the test9setup.sh script first as root.
# For test 10, run the test10setup.sh script between every trial.
i=1
while [ $i -lt 1000 ];do
    cp copy1.txt copy2.txt
    let i++
done
```

test11and12.sh

```
#!/bin/bash
# For test 11, create 1000 files first by running the test11setup.sh script
# For test 12, run the test12setup script as root between every trial.
i=1
while [ $i -lt 1000 ];do
    cp test11-file$i.txt test11-file-copy$i.txt
    let i++
done
```

test1setup.sh

```
#!/bin/bash
```

```
# Run this script as root to create the file to be read for test 1

echo “creating test 1 file...”
python -c “print 'G'*1000” >test1message.txt


echo “setting time attributes for test11 files...”
modtime -W 1 test1message.txt
chmod 777 test1message.txt
echo “done!”
```

test2setup.sh

```
#!/bin/bash
# Run this script as root between trials for test 2

echo “removing tifps attributes from test file...”
modtime -x test1message.txt
echo “done!”
```

test3setup.sh

```
#!/bin/bash
# Creates 1000 files for test 3, run this script as root
echo “creating 1000 test 3 files...”
i=1
while [ $i -lt 1000 ];do
    python -c “print 'G'*1000” >test3-file$i.txt
    let i++
done

echo “setting time attributes for test 3 files...”
modtime -W 1 test3-file* >/dev/null
chmod 777 test3-file*
echo “done!”
```

test4setup.sh

```
#!/bin/bash
# sets up files for test 4, run this script as root between trials

echo “removing time attributes for test 4 files...”
modtime -x test3-file* >/dev/null
echo “done!”
```

test5setup.sh

```
#!/bin/bash
# sets up the file for test 5, run this script as root

echo “Creating writefile.txt for test 5”
touch writefile.txt
chmod 777 writefile.txt
modtime -W 1 writefile.txt
echo “done!”
```

test6setup.sh

```
#!/bin/bash
# sets up the file for test 6, run this script as root between trials

echo “Removing tifps attributes for test 6”
modtime -x writefile.txt
echo “done!”
```

test7setup.sh

```
#!/bin/bash
# sets up the files for test 7, run this script as root
```

```
echo “Creating files for test 7”
i=1
while [ $i -lt 1000 ]; do
    touch test7-8-file$i.txt
    let i++
done

echo “setting tifps attributes”
chmod 777 test7-8-file*
modtime -W 1 test7-8-file* >/dev/null
echo “done!”
```

test9setup.sh

```
#!/bin/bash
# sets up the file for test 9, run this script as root

echo “Creating file for test 9”
python -c “print 'G'*1000”>copy1.txt

echo “setting tifps attributes on file”
chmod 777 copy1.txt
modtime -W 1 copy1.txt >/dev/null
echo “done!”
```

test10setup.sh

```
#!/bin/bash
# sets up the file for test 10, run this script as root for
# every trial run

echo “Removing time attributes for test 10”
modtime -x copy1.txt copy2.txt
echo “done!”
```

test11setup.sh

```
#!/bin/bash
# Creates 1000 files for test 11, run this script as root

echo “creating test 11 files...”
i=1
while [ $i -lt 1000 ];do
    python -c “print 'G'*1000” >test11-file$i.txt
    cp test11-file$i.txt test11-file-copy$i.txt
    let i++
done

echo “setting time attributes for test11 files...”
modtime -W 1 test11-file* >/dev/null
chmod 777 test11-file*
echo “done!”
```

test12setup.sh

```
#!/bin/bash
# setup file for test 12, run this script as root between trials

echo “deleting copies from test 11, if they exist...”
rm -f test11-file-copy*

echo “removing time attributes for test12 files...”
modtime -x test11-file* >/dev/null
chmod 777 test11-file*
echo “done!”
```

F. PERFORMANCE TEST RESULTS

This section contains the raw results of the tests described in Section D.

TIFPS Performance Tests **Last modified** 09/05/06

Test Environment: Dell Desktop 3.0 GHz, 256M Ram, Vmware server 1.0.0 image running FC5

Kernel revision #65 tested

		Tests											
Kernel:		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Normal		4.39	4.38	4.47	4.38	26.44	26.61	27.55	26.84	6.47	6.46	6.72	6.8
		4.4	4.42	4.47	4.42	26.67	26.69	27.5	27.17	6.55	6.38	6.77	6.87
		4.44	4.38	4.48	4.4	27.19	26.38	27.7	27.01	6.47	6.41	6.64	6.88
									27.17				
	Avg	**4.41**	**4.39**	**4.47**	**4.40**	**26.77**	**26.56**	**27.58**	**27.05**	**6.50**	**6.42**	**6.71**	**6.85**
	Std	**0.03**	**0.02**	**0.01**	**0.02**	**0.38**	**0.16**	**0.10**	**0.16**	**0.05**	**0.04**	**0.07**	**0.04**
TIFPS LSM		4.62	4.62	4.76	4.64	32.32	31.85	32.64	31.95	7.01	7.11	7.22	7.42
		4.68	4.56	4.7	4.64	32.67	31.72	32.05	32.5	7.15	7.08	7.17	7.4
		4.65	4.6	4.7	4.67	31.85	32.15	33.09	32.16	7.11	7.07	7.36	7.37
	Avg	**4.65**	**4.59**	**4.72**	**4.65**	**32.28**	**31.91**	**32.59**	**32.20**	**7.09**	**7.09**	**7.25**	**7.40**
	Std	**0.03**	**0.03**	**0.03**	**0.02**	**0.41**	**0.22**	**0.52**	**0.28**	**0.07**	**0.02**	**0.10**	**0.03**
Difference		5.4%	4.6%	5.5%	5.7%	20.6%	20.1%	18.2%	19.1%	9.1%	10.4%	8.0%	8.0%

Note: only sys time captured because it is time spent in the kernel; that's where the access control happens.

Tests:

1. Read 1 file with tifps attributes 1000 times; pipe output to /dev/null
2. Same as 1, but file does not have tifps attributes.
3. Read 1000 different files, each with tifps attributes
4. Same as 3, but files do not have tifps attributes.
5. Write to 1 file with tifps attribute 1000 times; “python -c “print 'G'*1000”>writefile.txt.
6. Same as 5, but file does not have tifps attributes.
7. Write to 1000 different existing files with tifps attributes
8. same as 7, except we remove all files created for each run (no tifps attributes)
9. Copy 1 file with tifps attributes 1000 times to another file also with tifps attributes (cp copy1.txt copy2.txt)
10. Same as 9, but source and dest files do not have tifps attibutes
11. Copy 1000 different files with tifps attributes to another 1000 set of files, also with tifps attributes.
12. Same as 11, but no source files have tifps attributes

Note: All units in are seconds unless otherwise noted.

G. CONCURRENCY TEST PROCEDURES

To test the robustness of the TIFPS LSM in multi-user concurrent access environments, the following test procedures were developed. To setup the concurrency tests:

Start a ***root*** login sessions, change to the **testscripts/concurrency/** directory and copy the ***-user.sh** (**3users-read-user.sh, 3users-write-user.sh, 3users-writedir-user.sh, 3users-copy-user.sh**) scripts to the **/bin** directoriy.

 # cp 3users--user.sh /bin/*

Concurrent read from and write to the same file

1. Start three additional login sessions; login to each session as ***root.***.
2. In the first ***root*** login session that is in the **testscripts/concurrency/** directory, run:

   # ./3users-read.sh

3. Within one minute, use the other three login sessions to login as ***sam, don, and jody*** to run their respective test scripts:

   # su - <user>

   $ 3users-read-user.sh

4. Record the results of the 4 login sessions from the screen and compare with the expected results listed below.
5. Clean up the test environment by running in the ***root*** session:

   # modtime –x /home/sam/.bash /home/jody/.bash* /home/don/.bash* /tmp*

6. Repeat steps 2 through 5 for the **3users-write.sh** and **3users-write-user.sh** scripts.
7. Expected results:
   a. Concurrent read from same file: The system should revoke access from the user when his/her time attribute expires.
   b. Concurrent write to same file: The system should do two things
      i. Transfer the time attributes of the most restrictive user to the shared file

ii. Revoke access for all users based on the the inherited time attributes.

Concurrent write to same directory and copy from same file

1. Exit the three user login sessions:

   $ exit

   # exit

2. In the remaining ***root*** login session, run:

   # modtime –x /home/jody/.bash /home/don/.bash* /home/sam/.bash* /tmp*

   # ./3users-writedir.sh

3. Record the results and compare the values on the screen with the expected results listed below.

4. Repeat step 2 with the commands:

   # modtime –x /home/jody/.bash /home/don/.bash* /home/sam/.bash* /tmp*

   # ./3users-copy.sh

5. Expected results:

   a. Concurrent write to same directory:

      i. Each copy written by the users should inherit the proper permissions from the users.

      ii. The time attributes of the directory should not change.

   b. Concurrent copy from the same file:

      i. Each copy of the file made by the user should take on the more restrictive of his user attributes or the the original file attributes.

      ii. The original file's time attributes should not change.

Table D-8. Summary of test scripts for concurrency testing

Test ID	Test Scripts	Description of concurrency test scenario	Expected results
G1	3users-read.sh 3users-read-user.sh	Concurrent read of a single file by 3 users with different time attributes	In each user's sessions, read access should be revoked at the time preset by ***root.***
G2	3users-write.sh 3users-write-user.sh	Concurrent write to a single file by 3 users with different time attributes	1. The file being written to should inherit the time attributes of the user with the most restrictive time attributes. 2. Each user's write access to that file should be revoked based upon the inherited time.
G3	3users-copy.sh 3users-copy-user.sh	Concurrent copy of a single file by 3 users with different time attributes	1. Each copy made by each user should inherit the user's time attributes 2. The directory's time attributes should not change.
G4	3users-writedir.sh 3users-writedir-user.sh	Concurrent write to a shared directory by 3 users with different time attributes	1. Each copy made by each user should inherit the more restrictive time attributes of either the user or the original file. 2. The destination directory's time attributes should not change.

H. CONCURRENCY TEST SCRIPTS

The scripts in this section correspond to tests described in Section G.

3users-read.sh

```
#!/bin/bash
# Run this script as root to set things up, then login as sam, don, and jody.
# As each of these users, run the "3user-read-user.sh" for each user within 1 minute.

echo "Concurrent read access test - multiple users"
echo ""

echo "Setting up sam, don, and jody's time attributes..."
modtime -M 1 -S15 /home/sam/.bash_profile
modtime -M 1 -S20 /home/don/.bash_profile
modtime -M 1 -S30 /home/jody/.bash_profile

echo "Setting up the object file to be read..."
echo "this message will self destruct in 10s" > /tmp/message.txt
modtime -W 1 /tmp/message.txt

echo ""
echo ""
echo "login as sam, don, and jody and run the 3user-read-user.sh script for each"
echo " within the next 1 Minute"
```

3users-read-user.sh

```
#!/bin/bash
# Run the 3user-read.sh script first as root, then
# within 60s, login with sam, don, and jody's accounts
# and run this script with each of these accounts.

echo "reading the /tmp/message.txt file continuously..."
echo ""
cat /tmp/message.txt

while [ 1 ]; do
    cat /tmp/message.txt >/dev/null
```

```
    if [ $? -gt 0 ]; then
        echo “Read access to /tmp/message.txt revoked, time:”
        date
        echo ““
        exit
    fi
done
```

3users-write.sh

```
#!/bin/bash
# Run this script as root to set things up, then login as sam, don, and jody.
# As each of these users, run the “3user-write-user.sh” for each user within 1 minute.

echo “Concurrent write access test by multiple users”
echo ““

echo “Setting up sam, don, and jody's time attributes...”
modtime -M 1 -S15 /home/sam/.bash_profile
modtime -M 1 -S20 /home/don/.bash_profile
modtime -M 1 -S30 /home/jody/.bash_profile

echo “Setting up the object file to be written to...”
echo “overwrite me” > /tmp/shared-write.txt
chmod 777 /tmp/shared-write.txt
modtime -W 1 /tmp/shared-write.txt

echo ““
echo ““
echo “login as sam, don, and jody and run the 3user-write-user.sh script within the next
minute”
```

3users-write-user.sh

```
#!/bin/bash
# Run the 3user-write.sh script first as root, then
# within 60s, login with sam, don, and jody's accounts
# and run this script with each of these accounts.

echo “attempting to write to /tmp/shared-write.txt file continuously...”
echo ““

while [ 1 ]; do
    echo “`date +%T`: $USER” >>/tmp/shared-write.txt
    if [ $? -gt 0 ]; then
        echo “write to /tmp/shared-write.txt failed, time:”
        date
        echo ““
        exit
    fi
done
```

3users-writedir.sh

```
#!/bin/bash
# Run this script as root; the 3users-writedir-user.sh script must be
# in the execute path for each of the 3 users sam, don, and jody.

echo “Test concurrent copying into same directory by multiple users”
echo ““

echo “Setting up sam, don, and jody's time attributes...”
modtime -M 1 -S15 /home/sam/.bash_profile
modtime -M 1 -S20 /home/don/.bash_profile
modtime -M 1 -S30 /home/jody/.bash_profile

echo “Setting up the object directory to be written...”
modtime -W 1 /tmp
```

```
echo “”
echo “”
rm -f /tmp/sam-copy.txt /tmp/don-copy.txt /tmp/jody-copy.txt
rm -f /home/sam/sam-copy.txt /home/don/don-copy.txt /home/jody/jody-copy.txt

su - sam -c “3users-writedir-user.sh” &
su - don -c “3users-writedir-user.sh” &
su - jody -c “3users-writedir-user.sh” &


sleep 10s
echo “The time attributes of the resulting copies made by each user are:”
modtime -g /tmp/sam-copy.txt
modtime -g /tmp/don-copy.txt
modtime -g /tmp/jody-copy.txt

echo “The time attribute of the original directory written:”
modtime -g /tmp
```

3users-writedir-user.sh

```
#!/bin/bash
# This script is run from the 3user-writedir.sh script by root

echo “$USER copying his/her respective private file”
echo “ continuously 1000 times to the /tmp directory”
echo “”

echo “$USER was here...” >$USER-copy.txt

i=0
while [ $i -lt 1000 ]; do
    cp $USER-copy.txt /tmp/
    let “i=$i+1”
done
```

3users-copy.sh

```
#!/bin/bash
# Run this script as root; the 3user-copy-user.sh script must be
# in the execute path for each of the 3 users sam, don, and jody.

echo “Concurrent file copy test by multiple users”
echo “”

echo “Setting up sam, don, and jody's time attributes...”
modtime -M 1 -S15 /home/sam/.bash_profile
modtime -M 1 -S20 /home/don/.bash_profile
modtime -M 1 -S30 /home/jody/.bash_profile

echo “Setting up the object file to be copied...”
echo “this message will self destruct in 10s” > /tmp/message.txt
modtime -W 1 /tmp/message.txt

echo “”
echo “”
rm -f /home/sam/sam-copy.txt
rm -f /home/don/don-copy.txt
rm -f /home/jody/jody-copy.txt
su - sam -c “3users-copy-user.sh” &
su - don -c “3users-copy-user.sh” &
su - jody -c “3users-copy-user.sh” &


sleep 10s
echo “The time attributes of the resulting copies made by each user are:”
modtime -g /home/sam/sam-message-copy.txt
modtime -g /home/don/don-message-copy.txt
```

```
modtime -g /home/jody/jody-message-copy.txt

echo “The time attribute of the original file copied:”
modtime -g /tmp/message.txt
```

3users-copy-user.sh

```
#!/bin/bash
# This script is run by the 3users-copy.sh script by root.

echo “$USER copying the /tmp/message.txt file continuously 1000 times”
echo ““

i=0
while [ $i -lt 1000 ]; do
    cp /tmp/message.txt $USER-message-copy.txt
    let “i=$i+1”
done
```

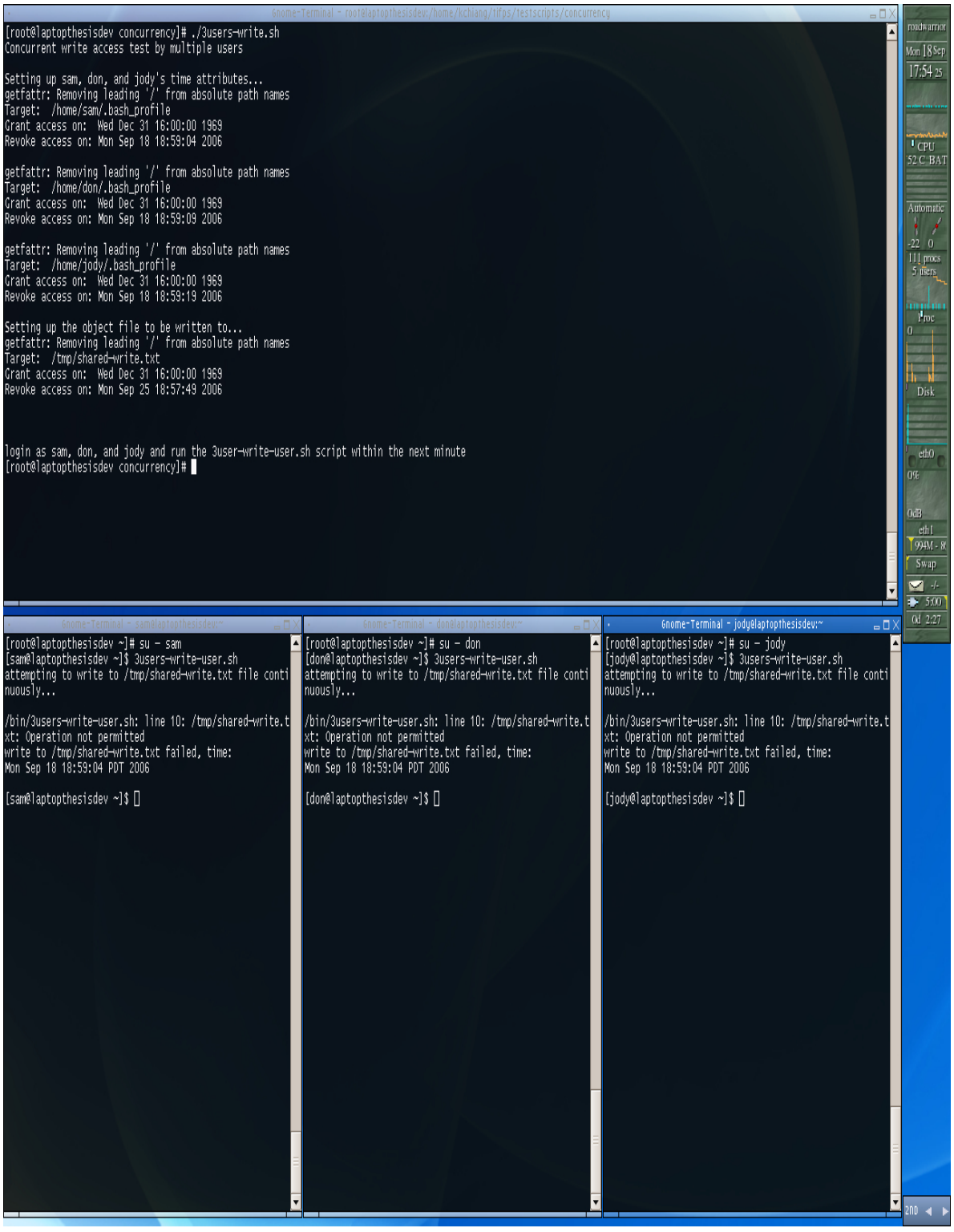
I. CONCURRENCY TEST RESULTS

This section contains results from the tests described in Section G.

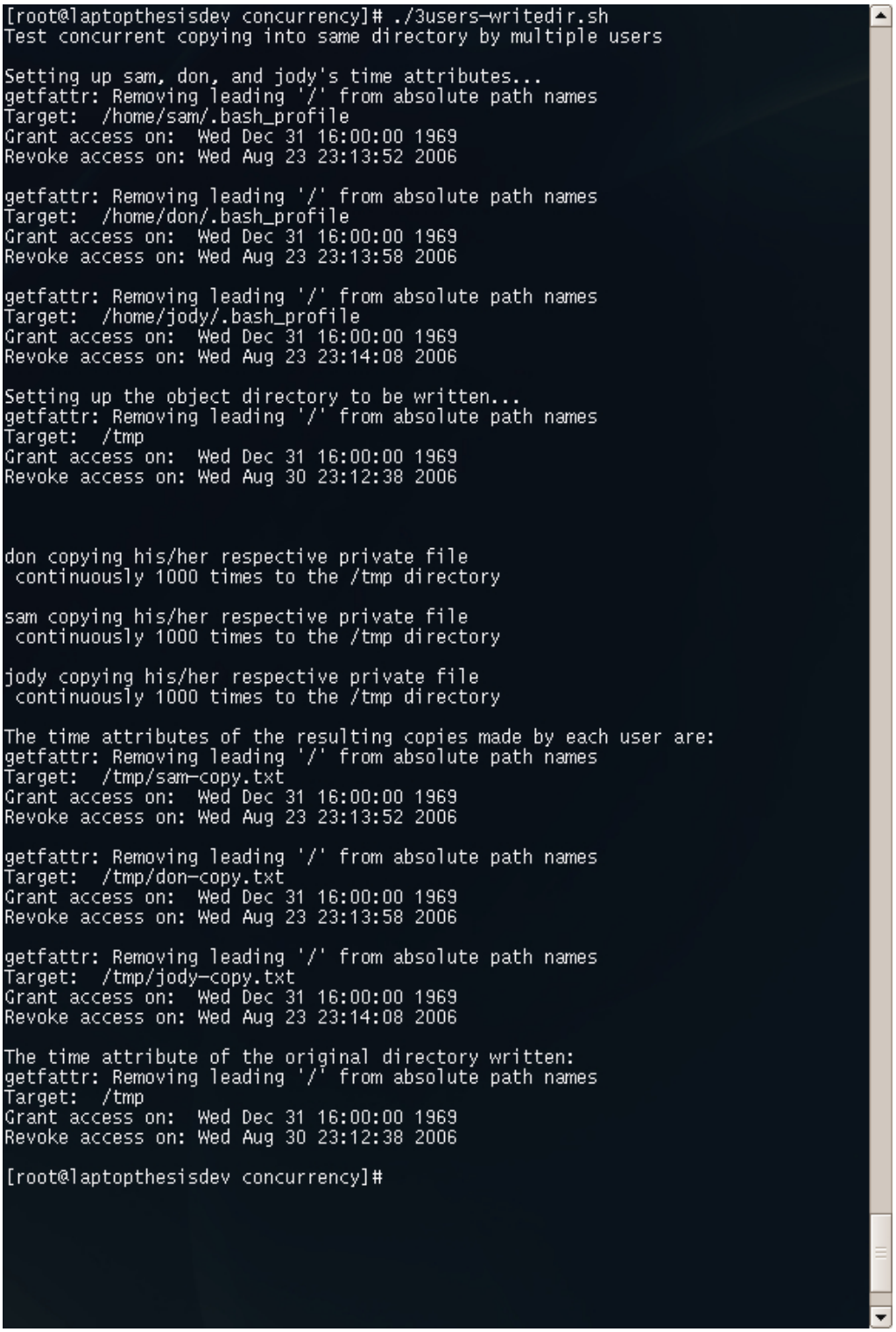
Concurrent read from the same file


Gnome-Terminal - root@laptopthesisdev:/home/kchlang/tlfps/testscripts/concurrency
[root@laptopthesisdev concurrency]# ./3users-read.sh
Concurrent read access test - multiple users
Setting up sam, don, and jody's time attributes...
getfattr: Removing leading '/' from absolute path names
Target: /home/sam/.bash_profile
Grant access on: Wed Dec 31 16:00:00 1969
Revoke access on: Mon Sep 18 18:57:11 2006
getfattr: Removing leading '/' from absolute path names
Target: /home/don/.bash_profile
Grant access on: Wed Dec 31 16:00:00 1969
Revoke access on: Mon Sep 18 18:57:16 2006
getfattr: Removing leading '/' from absolute path names
Target: /home/jody/.bash_profile
Grant access on: Wed Dec 31 16:00:00 1969
Revoke access on: Mon Sep 18 18:57:26 2006
Setting up the object file to be read...
getfattr: Removing leading '/' from absolute path names
Target: /tmp/message.txt
Grant access on: Sun Sep 10 19:29:17 2006
Revoke access on: Mon Sep 25 18:55:56 2006
login as sam, don, and jody and run the 3user-read-user.sh script for each
within the next 1 Minute
[root@laptopthesisdev concurrency]#
roadwarrior
Mon 18 Sep
17:50 37
CPU
62 C BAT
Automatic
-22 -22
114 procs
5 users
Proc
1.6M
Disk
eth0
0K
0dB
eth1
66M free
Swap
-/-
5000
Cal 2:23
Gnome-Terminal - sam@laptopthesisdev:~
[root@laptopthesisdev ~]# su - sam
[sam@laptopthesisdev ~]$ 3users-read-user.sh
reading the /tmp/message.txt file continuously...
this message will self destruct in 10s
cat: /tmp/message.txt: Operation not permitted
Read access to /tmp/message.txt revoked, time:
Mon Sep 18 18:57:11 PDT 2006
[sam@laptopthesisdev ~]$
Gnome-Terminal - don@laptopthesisdev:~
[root@laptopthesisdev ~]# su - don
[don@laptopthesisdev ~]$ 3users-read-user.sh
reading the /tmp/message.txt file continuously...
this message will self destruct in 10s
cat: /tmp/message.txt: Operation not permitted
Read access to /tmp/message.txt revoked, time:
Mon Sep 18 18:57:16 PDT 2006
[don@laptopthesisdev ~]$
Gnome-Terminal - jody@laptopthesisdev:~
[root@laptopthesisdev ~]# su - jody
[jody@laptopthesisdev ~]$ 3users-read-user.sh
reading the /tmp/message.txt file continuously...
this message will self destruct in 10s
cat: /tmp/message.txt: Operation not permitted
Read access to /tmp/message.txt revoked, time:
Mon Sep 18 18:57:26 PDT 2006
[jody@laptopthesisdev ~]$
2no

Concurrent write to same file



Concurrent write to same directory



Concurrent copy from the same file

```
[root@laptopthesisdev concurrency]# ./3users-copy.sh
Concurrent file copy test by multiple users

Setting up sam, don, and jody's time attributes...
getfattr: Removing leading '/' from absolute path names
Target:  /home/sam/.bash_profile
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Wed Aug 23 22:48:42 2006

getfattr: Removing leading '/' from absolute path names
Target:  /home/don/.bash_profile
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Wed Aug 23 22:48:48 2006

getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/.bash_profile
Grant access on:  Wed Dec 31 16:00:00 1969
Revoke access on: Wed Aug 23 22:48:58 2006

Setting up the object file to be copied...
getfattr: Removing leading '/' from absolute path names
Target:  /tmp/message.txt
Grant access on:  Wed Aug 23 00:14:35 2006
Revoke access on: Wed Aug 30 22:47:28 2006



sam copying the /tmp/message.txt file continuously 1000 times

jody copying the /tmp/message.txt file continuously 1000 times

don copying the /tmp/message.txt file continuously 1000 times

The time attributes of the resulting copies made by each user are:
getfattr: Removing leading '/' from absolute path names
Target:  /home/sam/sam-message-copy.txt
Grant access on:  Wed Aug 23 00:14:35 2006
Revoke access on: Wed Aug 23 22:48:42 2006

getfattr: Removing leading '/' from absolute path names
Target:  /home/don/don-message-copy.txt
Grant access on:  Wed Aug 23 00:14:35 2006
Revoke access on: Wed Aug 23 22:48:48 2006

getfattr: Removing leading '/' from absolute path names
Target:  /home/jody/jody-message-copy.txt
Grant access on:  Wed Aug 23 00:14:35 2006
Revoke access on: Wed Aug 23 22:48:58 2006

The time attribute of the original file copied:
getfattr: Removing leading '/' from absolute path names
Target:  /tmp/message.txt
Grant access on:  Wed Aug 23 00:14:35 2006
Revoke access on: Wed Aug 30 22:47:28 2006

[root@laptopthesisdev concurrency]#
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. DEVELOPMENT CONFIGURATION FILES

This appendix contains configuration files used during the development and testing of the TIFPS LSM. The kernel configuration file **.config** and the **emacs** editor configuration file **.emacs** are provided.

A. KERNEL .CONFIG CONFIGURATION FILE

```
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.15-tifps-082406-module
# Wed Aug 30 00:27:05 2006
#
CONFIG_X86_32=y
CONFIG_SEMAPHORE_SLEEPERS=y
CONFIG_X86=y
CONFIG_MMU=y
CONFIG_GENERIC_ISA_DMA=y
CONFIG_GENERIC_IOMAP=y
CONFIG_ARCH_MAY_HAVE_PC_FDC=y
CONFIG_DMI=y

#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
CONFIG_LOCK_KERNEL=y
CONFIG_INIT_ENV_ARG_LIMIT=32

#
# General setup
#
CONFIG_LOCALVERSION=“”
# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_SWAP=y
CONFIG_SYSVIPC=y
CONFIG_POSIX_MQUEUE=y
CONFIG_BSD_PROCESS_ACCT=y
# CONFIG_BSD_PROCESS_ACCT_V3 is not set
CONFIG_SYSCTL=y
CONFIG_AUDIT=y
CONFIG_AUDITSYSCALL=y
# CONFIG_IKCONFIG is not set
CONFIG_CPUSETS=y
CONFIG_INITRAMFS_SOURCE=“”
CONFIG_UID16=y
CONFIG_VM86=y
CONFIG_CC_OPTIMIZE_FOR_SIZE=y
# CONFIG_EMBEDDED is not set
CONFIG_KALLSYMS=y
# CONFIG_KALLSYMS_ALL is not set
CONFIG_KALLSYMS_EXTRA_PASS=y
CONFIG_HOTPLUG=y
CONFIG_PRINTK=y
CONFIG_BUG=y
CONFIG_ELF_CORE=y
CONFIG_BASE_FULL=y
CONFIG_FUTEX=y
CONFIG_EPOLL=y
CONFIG_SHMEM=y
CONFIG_CC_ALIGN_FUNCTIONS=0
CONFIG_CC_ALIGN_LABELS=0
CONFIG_CC_ALIGN_LOOPS=0
CONFIG_CC_ALIGN_JUMPS=0
CONFIG_SLAB=y
```

```
# CONFIG_TINY_SHMEM is not set
CONFIG_BASE_SMALL=0
# CONFIG_SLOB is not set

#
# Loadable module support
#
CONFIG_MODULES=y
CONFIG_MODULE_UNLOAD=y
# CONFIG_MODULE_FORCE_UNLOAD is not set
CONFIG_OBSOLETE_MODPARM=y
CONFIG_MODVERSIONS=y
CONFIG_MODULE_SRCVERSION_ALL=y
CONFIG_MODULE_SIG=y
# CONFIG_MODULE_SIG_FORCE is not set
CONFIG_KMOD=y
CONFIG_STOP_MACHINE=y

#
# Block layer
#
CONFIG_LBD=y

#
# IO Schedulers
#
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y
# CONFIG_DEFAULT_AS is not set
# CONFIG_DEFAULT_DEADLINE is not set
CONFIG_DEFAULT_CFQ=y
# CONFIG_DEFAULT_NOOP is not set
CONFIG_DEFAULT_IOSCHED="cfq"

#
# Processor type and features
#
CONFIG_X86_PC=y
# CONFIG_X86_XEN is not set
# CONFIG_X86_ELAN is not set
# CONFIG_X86_VOYAGER is not set
# CONFIG_X86_NUMAQ is not set
# CONFIG_X86_SUMMIT is not set
# CONFIG_X86_BIGSMP is not set
# CONFIG_X86_VISWS is not set
# CONFIG_X86_GENERICARCH is not set
# CONFIG_X86_ES7000 is not set
# CONFIG_M386 is not set
# CONFIG_M486 is not set
# CONFIG_M586 is not set
# CONFIG_M586TSC is not set
# CONFIG_M586MMX is not set
CONFIG_M686=y
# CONFIG_MPENTIUMII is not set
# CONFIG_MPENTIUMIII is not set
# CONFIG_MPENTIUMM is not set
# CONFIG_MPENTIUM4 is not set
# CONFIG_MK6 is not set
# CONFIG_MK7 is not set
# CONFIG_MK8 is not set
# CONFIG_MCRUSOE is not set
# CONFIG_MEFFICEON is not set
# CONFIG_MWINCHIPC6 is not set
# CONFIG_MWINCHIP2 is not set
# CONFIG_MWINCHIP3D is not set
# CONFIG_MGEODEGX1 is not set
# CONFIG_MGEODE_LX is not set
# CONFIG_MCYRIXIII is not set
# CONFIG_MVIAC3_2 is not set
```

```
# CONFIG_X86_GENERIC is not set
CONFIG_X86_CMPXCHG=y
CONFIG_X86_XADD=y
CONFIG_X86_L1_CACHE_SHIFT=5
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_GENERIC_CALIBRATE_DELAY=y
CONFIG_X86_PPRO_FENCE=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_X86_CMPXCHG64=y
CONFIG_X86_GOOD_APIC=y
CONFIG_X86_USE_PPRO_CHECKSUM=y
CONFIG_X86_TSC=y
CONFIG_HPET_TIMER=y
CONFIG_HPET_EMULATE_RTC=y
CONFIG_SMP=y
# CONFIG_SMP_ALTERNATIVES is not set
CONFIG_NR_CPUS=255
CONFIG_SCHED_SMT=y
# CONFIG_PREEMPT_NONE is not set
CONFIG_PREEMPT_VOLUNTARY=y
# CONFIG_PREEMPT is not set
CONFIG_PREEMPT_BKL=y
CONFIG_X86_LOCAL_APIC=y
CONFIG_X86_IO_APIC=y
CONFIG_X86_MCE=y
# CONFIG_X86_MCE_NONFATAL is not set
# CONFIG_X86_MCE_P4THERMAL is not set
# CONFIG_TOSHIBA is not set
# CONFIG_I8K is not set
# CONFIG_X86_REBOOTFIXUPS is not set
CONFIG_MICROCODE=m
CONFIG_X86_MSR=y
CONFIG_X86_CPUID=y
# CONFIG_SWIOTLB is not set

#
# Firmware Drivers
#
CONFIG_EDD=m
CONFIG_DELL_RBU=m
CONFIG_DCDBAS=m
CONFIG_NOHIGHMEM=y
# CONFIG_HIGHMEM4G is not set
# CONFIG_HIGHMEM64G is not set
CONFIG_VMSPLIT_3G=y
# CONFIG_VMSPLIT_3G_OPT is not set
# CONFIG_VMSPLIT_2G is not set
# CONFIG_VMSPLIT_1G is not set
CONFIG_PAGE_OFFSET=0xC0000000
CONFIG_ARCH_FLATMEM_ENABLE=y
CONFIG_ARCH_SPARSEMEM_ENABLE=y
CONFIG_ARCH_SELECT_MEMORY_MODEL=y
CONFIG_SELECT_MEMORY_MODEL=y
CONFIG_FLATMEM_MANUAL=y
# CONFIG_DISCONTIGMEM_MANUAL is not set
# CONFIG_SPARSEMEM_MANUAL is not set
CONFIG_FLATMEM=y
CONFIG_FLAT_NODE_MEM_MAP=y
CONFIG_SPARSEMEM_STATIC=y
CONFIG_SPLIT_PTLOCK_CPUS=4
# CONFIG_MATH_EMULATION is not set
CONFIG_MTRR=y
# CONFIG_EFI is not set
CONFIG_IRQBALANCE=y
# CONFIG_REGPARM is not set
# CONFIG_SECCOMP is not set
# CONFIG_HZ_100 is not set
CONFIG_HZ_250=y
```

```
# CONFIG_HZ_1000 is not set
CONFIG_HZ=250
CONFIG_KEXEC=y
CONFIG_PHYSICAL_START=0x100000
# CONFIG_HOTPLUG_CPU is not set
CONFIG_DOUBLEFAULT=y

#
# Power management options (ACPI, APM)
#
CONFIG_PM=y
# CONFIG_PM_LEGACY is not set
# CONFIG_PM_DEBUG is not set

#
# ACPI (Advanced Configuration and Power Interface) Support
#
CONFIG_ACPI=y
CONFIG_ACPI_AC=m
CONFIG_ACPI_BATTERY=m
CONFIG_ACPI_BUTTON=m
CONFIG_ACPI_VIDEO=m
# CONFIG_ACPI_HOTKEY is not set
CONFIG_ACPI_FAN=y
CONFIG_ACPI_PROCESSOR=y
CONFIG_ACPI_THERMAL=y
# CONFIG_ACPI_ASUS is not set
# CONFIG_ACPI_IBM is not set
# CONFIG_ACPI_TOSHIBA is not set
CONFIG_ACPI_BLACKLIST_YEAR=0
# CONFIG_ACPI_DEBUG is not set
CONFIG_ACPI_EC=y
CONFIG_ACPI_POWER=y
CONFIG_ACPI_SYSTEM=y
CONFIG_X86_PM_TIMER=y
CONFIG_ACPI_CONTAINER=y

#
# APM (Advanced Power Management) BIOS Support
#

#
# CPU Frequency scaling
#
# CONFIG_CPU_FREQ is not set

#
# Bus options (PCI, PCMCIA, EISA, MCA, ISA)
#
CONFIG_PCI=y
# CONFIG_PCI_GOBIOS is not set
# CONFIG_PCI_GOMMCONFIG is not set
# CONFIG_PCI_GODIRECT is not set
# CONFIG_PCI_GOXEN_FE is not set
CONFIG_PCI_GOANY=y
CONFIG_PCI_BIOS=y
CONFIG_PCI_DIRECT=y
CONFIG_PCI_MMCONFIG=y
CONFIG_PCIEPORTBUS=y
# CONFIG_PCI_MSI is not set
CONFIG_PCI_LEGACY_PROC=y
# CONFIG_PCI_DEBUG is not set
CONFIG_ISA_DMA_API=y
# CONFIG_ISA is not set
# CONFIG_MCA is not set
# CONFIG_SCx200 is not set

#
# PCCARD (PCMCIA/CardBus) support
#
CONFIG_PCCARD=y
```

```
# CONFIG_PCMCIA_DEBUG is not set
CONFIG_PCMCIA=y
CONFIG_PCMCIA_LOAD_CIS=y
CONFIG_PCMCIA_IOCTL=y
CONFIG_CARDBUS=y

#
# PC-card bridges
#
# CONFIG_YENTA is not set
# CONFIG_PD6729 is not set
# CONFIG_I82092 is not set

#
# PCI Hotplug Support
#
# CONFIG_HOTPLUG_PCI is not set

#
# Executable file formats
#
CONFIG_BINFMT_ELF=y
# CONFIG_BINFMT_AOUT is not set
CONFIG_BINFMT_MISC=y

#
# Networking
#
CONFIG_NET=y

#
# Networking options
#
# CONFIG_NETDEBUG is not set
CONFIG_PACKET=y
CONFIG_PACKET_MMAP=y
CONFIG_UNIX=y
CONFIG_XFRM=y
CONFIG_XFRM_USER=y
CONFIG_NET_KEY=y
CONFIG_INET=y
CONFIG_IP_MULTICAST=y
CONFIG_IP_ADVANCED_ROUTER=y
CONFIG_ASK_IP_FIB_HASH=y
# CONFIG_IP_FIB_TRIE is not set
CONFIG_IP_FIB_HASH=y
CONFIG_IP_MULTIPLE_TABLES=y
CONFIG_IP_ROUTE_MULTIPATH=y
# CONFIG_IP_ROUTE_MULTIPATH_CACHED is not set
CONFIG_IP_ROUTE_VERBOSE=y
# CONFIG_IP_PNP is not set
# CONFIG_NET_IPIP is not set
# CONFIG_NET_IPGRE is not set
CONFIG_IP_MROUTE=y
CONFIG_IP_PIMSM_V1=y
CONFIG_IP_PIMSM_V2=y
# CONFIG_ARPD is not set
CONFIG_SYN_COOKIES=y
# CONFIG_INET_AH is not set
# CONFIG_INET_ESP is not set
# CONFIG_INET_IPCOMP is not set
# CONFIG_INET_TUNNEL is not set
# CONFIG_INET_DIAG is not set
# CONFIG_TCP_CONG_ADVANCED is not set
CONFIG_TCP_CONG_BIC=y
# CONFIG_IPV6 is not set
# CONFIG_NETFILTER is not set

#
# DCCP Configuration (EXPERIMENTAL)
#
```

```
# CONFIG_IP_DCCP is not set

#
# SCTP Configuration (EXPERIMENTAL)
#
# CONFIG_IP_SCTP is not set

#
# TIPC Configuration (EXPERIMENTAL)
#
# CONFIG_TIPC is not set
# CONFIG_ATM is not set
# CONFIG_BRIDGE is not set
# CONFIG_VLAN_8021Q is not set
# CONFIG_DECNET is not set
# CONFIG_LLC2 is not set
# CONFIG_IPX is not set
# CONFIG_ATALK is not set
# CONFIG_X25 is not set
# CONFIG_LAPB is not set
# CONFIG_NET_DIVERT is not set
# CONFIG_ECONET is not set
# CONFIG_WAN_ROUTER is not set

#
# QoS and/or fair queueing
#
# CONFIG_NET_SCHED is not set

#
# Network testing
#
# CONFIG_NET_PKTGEN is not set
# CONFIG_HAMRADIO is not set
# CONFIG_IRDA is not set
# CONFIG_BT is not set
# CONFIG_IEEE80211 is not set
# CONFIG_TUX is not set

#
# Device Drivers
#

#
# Generic Driver Options
#
CONFIG_STANDALONE=y
CONFIG_PREVENT_FIRMWARE_BUILD=y
CONFIG_FW_LOADER=y
# CONFIG_DEBUG_DRIVER is not set

#
# Connector - unified userspace <-> kernelspace linker
#
CONFIG_CONNECTOR=m

#
# Memory Technology Devices (MTD)
#
# CONFIG_MTD is not set

#
# Parallel port support
#
# CONFIG_PARPORT is not set

#
# Plug and Play support
#
CONFIG_PNP=y
# CONFIG_PNP_DEBUG is not set
```

```
#
# Protocols
#
CONFIG_PNPACPI=y

#
# Block devices
#
# CONFIG_BLK_DEV_FD is not set
# CONFIG_BLK_CPQ_DA is not set
# CONFIG_BLK_CPQ_CISS_DA is not set
# CONFIG_BLK_DEV_DAC960 is not set
# CONFIG_BLK_DEV_UMEM is not set
# CONFIG_BLK_DEV_COW_COMMON is not set
CONFIG_BLK_DEV_LOOP=m
CONFIG_BLK_DEV_CRYPTOLOOP=m
CONFIG_BLK_DEV_NBD=m
# CONFIG_BLK_DEV_SX8 is not set
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_COUNT=16
CONFIG_BLK_DEV_RAM_SIZE=16384
CONFIG_BLK_DEV_INITRD=y
CONFIG_CDROM_PKTCDVD=m
CONFIG_CDROM_PKTCDVD_BUFFERS=8
# CONFIG_CDROM_PKTCDVD_WCACHE is not set
CONFIG_DISKDUMP=m
CONFIG_ATA_OVER_ETH=m

#
# ATA/ATAPI/MFM/RLL support
#
CONFIG_IDE=y
CONFIG_BLK_DEV_IDE=y

#
# Please see Documentation/ide.txt for help/info on IDE drives
#
# CONFIG_BLK_DEV_IDE_SATA is not set
# CONFIG_BLK_DEV_HD_IDE is not set
CONFIG_BLK_DEV_IDEDISK=y
CONFIG_IDEDISK_MULTI_MODE=y
CONFIG_BLK_DEV_IDECS=m
CONFIG_BLK_DEV_IDECD=y
# CONFIG_BLK_DEV_IDETAPE is not set
CONFIG_BLK_DEV_IDEFLOPPY=y
CONFIG_BLK_DEV_IDESCSI=m
CONFIG_IDE_TASK_IOCTL=y

#
# IDE chipset support/bugfixes
#
CONFIG_IDE_GENERIC=y
CONFIG_BLK_DEV_CMD640=y
CONFIG_BLK_DEV_CMD640_ENHANCED=y
CONFIG_BLK_DEV_IDEPNP=y
CONFIG_BLK_DEV_IDEPCI=y
CONFIG_IDEPCI_SHARE_IRQ=y
# CONFIG_BLK_DEV_OFFBOARD is not set
CONFIG_BLK_DEV_GENERIC=y
# CONFIG_BLK_DEV_OPTI621 is not set
CONFIG_BLK_DEV_RZ1000=y
CONFIG_BLK_DEV_IDEDMA_PCI=y
# CONFIG_BLK_DEV_IDEDMA_FORCED is not set
CONFIG_IDEDMA_PCI_AUTO=y
# CONFIG_IDEDMA_ONLYDISK is not set
CONFIG_BLK_DEV_AEC62XX=y
CONFIG_BLK_DEV_ALI15X3=y
# CONFIG_WDC_ALI15X3 is not set
CONFIG_BLK_DEV_AMD74XX=y
CONFIG_BLK_DEV_ATIIXP=y
```

```
CONFIG_BLK_DEV_CMD64X=y
CONFIG_BLK_DEV_TRIFLEX=y
CONFIG_BLK_DEV_CY82C693=y
CONFIG_BLK_DEV_CS5520=y
CONFIG_BLK_DEV_CS5530=y
# CONFIG_BLK_DEV_CS5535 is not set
CONFIG_BLK_DEV_HPT34X=y
# CONFIG_HPT34X_AUTODMA is not set
CONFIG_BLK_DEV_HPT366=y
# CONFIG_BLK_DEV_SC1200 is not set
CONFIG_BLK_DEV_PIIX=y
CONFIG_BLK_DEV_IT821X=y
# CONFIG_BLK_DEV_NS87415 is not set
CONFIG_BLK_DEV_PDC202XX_OLD=y
# CONFIG_PDC202XX_BURST is not set
CONFIG_BLK_DEV_PDC202XX_NEW=y
CONFIG_BLK_DEV_SVWKS=y
CONFIG_BLK_DEV_SIIMAGE=y
CONFIG_BLK_DEV_SIS5513=y
CONFIG_BLK_DEV_SLC90E66=y
# CONFIG_BLK_DEV_TRM290 is not set
CONFIG_BLK_DEV_VIA82CXXX=y
# CONFIG_IDE_ARM is not set
CONFIG_BLK_DEV_IDEDMA=y
# CONFIG_IDEDMA_IVB is not set
CONFIG_IDEDMA_AUTO=y
# CONFIG_BLK_DEV_HD is not set

#
# SCSI device support
#
CONFIG_RAID_ATTRS=m
CONFIG_SCSI=m
CONFIG_SCSI_PROC_FS=y

#
# SCSI support type (disk, tape, CD-ROM)
#
CONFIG_BLK_DEV_SD=m
# CONFIG_CHR_DEV_ST is not set
# CONFIG_CHR_DEV_OSST is not set
CONFIG_BLK_DEV_SR=m
CONFIG_BLK_DEV_SR_VENDOR=y
CONFIG_CHR_DEV_SG=m
CONFIG_CHR_DEV_SCH=m

#
# Some SCSI devices (e.g. CD jukebox) support multiple LUNs
#
CONFIG_SCSI_MULTI_LUN=y
# CONFIG_SCSI_CONSTANTS is not set
CONFIG_SCSI_LOGGING=y

#
# SCSI Transport Attributes
#
CONFIG_SCSI_SPI_ATTRS=m
CONFIG_SCSI_FC_ATTRS=m
CONFIG_SCSI_ISCSI_ATTRS=m
CONFIG_SCSI_SAS_ATTRS=m

#
# SCSI low-level drivers
#
CONFIG_ISCSI_TCP=m
CONFIG_BLK_DEV_3W_XXXX_RAID=m
CONFIG_SCSI_3W_9XXX=m
CONFIG_SCSI_ACARD=m
CONFIG_SCSI_AACRAID=m
CONFIG_SCSI_AIC7XXX=m
CONFIG_AIC7XXX_CMDS_PER_DEVICE=4
```

```
CONFIG_AIC7XXX_RESET_DELAY_MS=15000
# CONFIG_AIC7XXX_DEBUG_ENABLE is not set
CONFIG_AIC7XXX_DEBUG_MASK=0
# CONFIG_AIC7XXX_REG_PRETTY_PRINT is not set
# CONFIG_SCSI_AIC7XXX_OLD is not set
# CONFIG_SCSI_AIC79XX is not set
# CONFIG_SCSI_DPT_I2O is not set
# CONFIG_SCSI_ADVANSYS is not set
CONFIG_MEGARAID_NEWGEN=y
CONFIG_MEGARAID_MM=m
CONFIG_MEGARAID_MAILBOX=m
# CONFIG_MEGARAID_LEGACY is not set
# CONFIG_MEGARAID_SAS is not set
# CONFIG_SCSI_SATA is not set
CONFIG_SCSI_BUSLOGIC=m
# CONFIG_SCSI_OMIT_FLASHPOINT is not set
# CONFIG_SCSI_DMX3191D is not set
# CONFIG_SCSI_EATA is not set
# CONFIG_SCSI_FUTURE_DOMAIN is not set
CONFIG_SCSI_GDTH=m
CONFIG_SCSI_IPS=m
CONFIG_SCSI_INITIO=m
CONFIG_SCSI_INIA100=m
CONFIG_SCSI_SYM53C8XX_2=m
CONFIG_SCSI_SYM53C8XX_DMA_ADDRESSING_MODE=1
CONFIG_SCSI_SYM53C8XX_DEFAULT_TAGS=16
CONFIG_SCSI_SYM53C8XX_MAX_TAGS=64
# CONFIG_SCSI_SYM53C8XX_IOMAPPED is not set
# CONFIG_SCSI_IPR is not set
# CONFIG_SCSI_QLOGIC_FC is not set
# CONFIG_SCSI_QLOGIC_1280 is not set
# CONFIG_SCSI_QLA_FC is not set
# CONFIG_SCSI_LPFC is not set
# CONFIG_SCSI_DC395x is not set
# CONFIG_SCSI_DC390T is not set
# CONFIG_SCSI_NSP32 is not set
# CONFIG_SCSI_DEBUG is not set

#
# PCMCIA SCSI adapter support
#
# CONFIG_PCMCIA_AHA152X is not set
# CONFIG_PCMCIA_FDOMAIN is not set
# CONFIG_PCMCIA_NINJA_SCSI is not set
CONFIG_PCMCIA_QLOGIC=m
CONFIG_PCMCIA_SYM53C500=m

#
# Multi-device support (RAID and LVM)
#
CONFIG_MD=y
CONFIG_BLK_DEV_MD=y
CONFIG_MD_LINEAR=m
CONFIG_MD_RAID0=m
CONFIG_MD_RAID1=m
CONFIG_MD_RAID10=m
CONFIG_MD_RAID5=m
CONFIG_MD_RAID6=m
CONFIG_MD_MULTIPATH=m
CONFIG_MD_FAULTY=m
CONFIG_BLK_DEV_DM=m
CONFIG_DM_CRYPT=m
CONFIG_DM_SNAPSHOT=m
CONFIG_DM_MIRROR=m
CONFIG_DM_ZERO=m
CONFIG_DM_MULTIPATH=m
CONFIG_DM_MULTIPATH_EMC=m

#
# Fusion MPT device support
#
```

```
CONFIG_FUSION=y
CONFIG_FUSION_SPI=m
CONFIG_FUSION_FC=m
CONFIG_FUSION_SAS=m
CONFIG_FUSION_MAX_SGE=40
CONFIG_FUSION_CTL=m

#
# IEEE 1394 (FireWire) support
#
# CONFIG_IEEE1394 is not set

#
# I2O device support
#
# CONFIG_I2O is not set

#
# Network device support
#
CONFIG_NETDEVICES=y
CONFIG_DUMMY=m
CONFIG_BONDING=m
CONFIG_EQUALIZER=m
CONFIG_TUN=m
CONFIG_NET_SB1000=m

#
# ARCnet devices
#
# CONFIG_ARCNET is not set

#
# PHY device support
#
CONFIG_PHYLIB=m

#
# MII PHY device drivers
#
CONFIG_MARVELL_PHY=m
CONFIG_DAVICOM_PHY=m
CONFIG_QSEMI_PHY=m
CONFIG_LXT_PHY=m
CONFIG_CICADA_PHY=m

#
# Ethernet (10 or 100Mbit)
#
CONFIG_NET_ETHERNET=y
CONFIG_MII=y
# CONFIG_HAPPYMEAL is not set
# CONFIG_SUNGEM is not set
# CONFIG_CASSINI is not set
# CONFIG_NET_VENDOR_3COM is not set

#
# Tulip family network device support
#
# CONFIG_NET_TULIP is not set
# CONFIG_HP100 is not set
CONFIG_NET_PCI=y
CONFIG_PCNET32=m
# CONFIG_AMD8111_ETH is not set
# CONFIG_ADAPTEC_STARFIRE is not set
# CONFIG_B44 is not set
# CONFIG_FORCEDETH is not set
# CONFIG_DGRS is not set
# CONFIG_EEPRO100 is not set
# CONFIG_E100 is not set
# CONFIG_FEALNX is not set
```

```
# CONFIG_NATSEMI is not set
# CONFIG_NE2K_PCI is not set
# CONFIG_8139CP is not set
# CONFIG_8139TOO is not set
# CONFIG_SIS900 is not set
# CONFIG_EPIC100 is not set
# CONFIG_SUNDANCE is not set
# CONFIG_TLAN is not set
# CONFIG_VIA_RHINE is not set

#
# Ethernet (1000 Mbit)
#
# CONFIG_ACENIC is not set
# CONFIG_DL2K is not set
# CONFIG_E1000 is not set
# CONFIG_NS83820 is not set
# CONFIG_HAMACHI is not set
# CONFIG_YELLOWFIN is not set
# CONFIG_R8169 is not set
# CONFIG_SIS190 is not set
# CONFIG_SKGE is not set
# CONFIG_SKY2 is not set
# CONFIG_SK98LIN is not set
# CONFIG_VIA_VELOCITY is not set
# CONFIG_TIGON3 is not set
# CONFIG_BNX2 is not set

#
# Ethernet (10000 Mbit)
#
# CONFIG_CHELSIO_T1 is not set
# CONFIG_IXGB is not set
# CONFIG_S2IO is not set

#
# Token Ring devices
#
# CONFIG_TR is not set

#
# Wireless LAN (non-hamradio)
#
# CONFIG_NET_RADIO is not set

#
# PCMCIA network device support
#
# CONFIG_NET_PCMCIA is not set

#
# Wan interfaces
#
# CONFIG_WAN is not set
# CONFIG_FDDI is not set
# CONFIG_HIPPI is not set
# CONFIG_PPP is not set
# CONFIG_SLIP is not set
# CONFIG_NET_FC is not set
# CONFIG_SHAPER is not set
# CONFIG_NETCONSOLE is not set
# CONFIG_NETPOLL is not set
# CONFIG_NET_POLL_CONTROLLER is not set

#
# ISDN subsystem
#
# CONFIG_ISDN is not set

#
# Telephony Support
```

```
#
# CONFIG_PHONE is not set

#
# Input device support
#
CONFIG_INPUT=y

#
# Userland interfaces
#
CONFIG_INPUT_MOUSEDEV=y
# CONFIG_INPUT_MOUSEDEV_PSAUX is not set
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
# CONFIG_INPUT_JOYDEV is not set
# CONFIG_INPUT_TSDEV is not set
CONFIG_INPUT_EVDEV=y
# CONFIG_INPUT_EVBUG is not set

#
# Input Device Drivers
#
CONFIG_INPUT_KEYBOARD=y
CONFIG_KEYBOARD_ATKBD=y
# CONFIG_KEYBOARD_SUNKBD is not set
# CONFIG_KEYBOARD_LKKBD is not set
# CONFIG_KEYBOARD_XTKBD is not set
# CONFIG_KEYBOARD_NEWTON is not set
CONFIG_INPUT_MOUSE=y
CONFIG_MOUSE_PS2=y
CONFIG_MOUSE_SERIAL=m
CONFIG_MOUSE_VSXXXAA=m
# CONFIG_INPUT_JOYSTICK is not set
# CONFIG_INPUT_TOUCHSCREEN is not set
# CONFIG_INPUT_MISC is not set

#
# Hardware I/O ports
#
CONFIG_SERIO=y
CONFIG_SERIO_I8042=y
CONFIG_SERIO_SERPORT=y
# CONFIG_SERIO_CT82C710 is not set
# CONFIG_SERIO_PCIPS2 is not set
CONFIG_SERIO_LIBPS2=y
# CONFIG_SERIO_RAW is not set
CONFIG_GAMEPORT=y
CONFIG_GAMEPORT_NS558=m
CONFIG_GAMEPORT_L4=m
CONFIG_GAMEPORT_EMU10K1=m
CONFIG_GAMEPORT_FM801=m

#
# Character devices
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_HW_CONSOLE=y
# CONFIG_SERIAL_NONSTANDARD is not set

#
# Serial drivers
#
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_CONSOLE=y
CONFIG_SERIAL_8250_CS=m
# CONFIG_SERIAL_8250_ACPI is not set
CONFIG_SERIAL_8250_NR_UARTS=32
CONFIG_SERIAL_8250_RUNTIME_UARTS=4
CONFIG_SERIAL_8250_EXTENDED=y
```

```
CONFIG_SERIAL_8250_MANY_PORTS=y
CONFIG_SERIAL_8250_SHARE_IRQ=y
CONFIG_SERIAL_8250_DETECT_IRQ=y
CONFIG_SERIAL_8250_RSA=y

#
# Non-8250 serial port support
#
CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_CORE_CONSOLE=y
# CONFIG_SERIAL_JSM is not set
CONFIG_UNIX98_PTYS=y
# CONFIG_LEGACY_PTYS is not set
CONFIG_CRASH=m

#
# IPMI
#
CONFIG_IPMI_HANDLER=m
# CONFIG_IPMI_PANIC_EVENT is not set
CONFIG_IPMI_DEVICE_INTERFACE=m
CONFIG_IPMI_SI=m
CONFIG_IPMI_WATCHDOG=m
CONFIG_IPMI_POWEROFF=m

#
# Watchdog Cards
#
# CONFIG_WATCHDOG is not set
CONFIG_HW_RANDOM=m
CONFIG_NVRAM=m
CONFIG_RTC=y
CONFIG_DTLK=m
CONFIG_R3964=m
# CONFIG_APPLICOM is not set
# CONFIG_SONYPI is not set

#
# Ftape, the floppy tape device driver
#
CONFIG_AGP=y
# CONFIG_AGP_ALI is not set
# CONFIG_AGP_ATI is not set
# CONFIG_AGP_AMD is not set
CONFIG_AGP_AMD64=y
CONFIG_AGP_INTEL=y
# CONFIG_AGP_NVIDIA is not set
# CONFIG_AGP_SIS is not set
# CONFIG_AGP_SWORKS is not set
# CONFIG_AGP_VIA is not set
# CONFIG_AGP_EFFICEON is not set
# CONFIG_DRM is not set

#
# PCMCIA character devices
#
# CONFIG_SYNCLINK_CS is not set
CONFIG_CARDMAN_4000=m
CONFIG_CARDMAN_4040=m
# CONFIG_MWAVE is not set
# CONFIG_CS5535_GPIO is not set
# CONFIG_RAW_DRIVER is not set
CONFIG_HPET=y
# CONFIG_HPET_RTC_IRQ is not set
# CONFIG_HPET_MMAP is not set
CONFIG_HANGCHECK_TIMER=m

#
# TPM devices
#
# CONFIG_TCG_TPM is not set
```

```
# CONFIG_TELCLOCK is not set

#
# I2C support
#
# CONFIG_I2C is not set

#
# SPI support
#
# CONFIG_SPI is not set
# CONFIG_SPI_MASTER is not set

#
# Dallas's 1-wire bus
#
# CONFIG_W1 is not set

#
# Hardware Monitoring support
#
# CONFIG_HWMON is not set
# CONFIG_HWMON_VID is not set

#
# Misc devices
#
# CONFIG_IBM_ASM is not set

#
# Multimedia Capabilities Port drivers
#

#
# Multimedia devices
#
# CONFIG_VIDEO_DEV is not set

#
# Digital Video Broadcasting Devices
#
# CONFIG_DVB is not set

#
# Graphics support
#
CONFIG_FB=y
CONFIG_FB_CFB_FILLRECT=y
CONFIG_FB_CFB_COPYAREA=y
CONFIG_FB_CFB_IMAGEBLIT=y
# CONFIG_FB_MACMODES is not set
CONFIG_FB_MODE_HELPERS=y
CONFIG_FB_TILEBLITTING=y
CONFIG_FB_CIRRUS=m
# CONFIG_FB_PM2 is not set
# CONFIG_FB_CYBER2000 is not set
# CONFIG_FB_ARC is not set
# CONFIG_FB_ASILIANT is not set
# CONFIG_FB_IMSTT is not set
CONFIG_FB_VGA16=m
CONFIG_FB_VESA=y
CONFIG_VIDEO_SELECT=y
# CONFIG_FB_HGA is not set
# CONFIG_FB_S1D13XXX is not set
# CONFIG_FB_NVIDIA is not set
# CONFIG_FB_RIVA is not set
# CONFIG_FB_I810 is not set
# CONFIG_FB_INTEL is not set
# CONFIG_FB_MATROX is not set
# CONFIG_FB_RADEON_OLD is not set
# CONFIG_FB_RADEON is not set
```

```
# CONFIG_FB_ATY128 is not set
# CONFIG_FB_ATY is not set
# CONFIG_FB_SAVAGE is not set
# CONFIG_FB_SIS is not set
# CONFIG_FB_NEOMAGIC is not set
# CONFIG_FB_KYRO is not set
# CONFIG_FB_3DFX is not set
# CONFIG_FB_VOODOO1 is not set
# CONFIG_FB_CYBLA is not set
# CONFIG_FB_TRIDENT is not set
# CONFIG_FB_GEODE is not set
# CONFIG_FB_VIRTUAL is not set

#
# Console display driver support
#
CONFIG_VGA_CONSOLE=y
CONFIG_DUMMY_CONSOLE=y
CONFIG_FRAMEBUFFER_CONSOLE=y
CONFIG_FRAMEBUFFER_CONSOLE_ROTATION=y
# CONFIG_FONTS is not set
CONFIG_FONT_8x8=y
CONFIG_FONT_8x16=y

#
# Logo configuration
#
CONFIG_LOGO=y
# CONFIG_LOGO_LINUX_MONO is not set
# CONFIG_LOGO_LINUX_VGA16 is not set
CONFIG_LOGO_LINUX_CLUT224=y
# CONFIG_BACKLIGHT_LCD_SUPPORT is not set

#
# Sound
#
# CONFIG_SOUND is not set

#
# USB support
#
CONFIG_USB_ARCH_HAS_HCD=y
CONFIG_USB_ARCH_HAS_OHCI=y
# CONFIG_USB is not set

#
# NOTE: USB_STORAGE enables SCSI, and 'SCSI disk support'
#

#
# USB Gadget Support
#
# CONFIG_USB_GADGET is not set

#
# MMC/SD Card support
#
# CONFIG_MMC is not set

#
# InfiniBand support
#
# CONFIG_INFINIBAND is not set

#
# EDAC - error detection and reporting (RAS) (EXPERIMENTAL)
#
CONFIG_EDAC=y

#
# Reporting subsystems
```

```
#
# CONFIG_EDAC_DEBUG is not set
CONFIG_EDAC_MM_EDAC=m
# CONFIG_EDAC_AMD76X is not set
CONFIG_EDAC_E7XXX=m
CONFIG_EDAC_E752X=m
CONFIG_EDAC_I82875P=m
CONFIG_EDAC_I82860=m
CONFIG_EDAC_R82600=m
CONFIG_EDAC_POLL=y

#
# File systems
#
CONFIG_EXT2_FS=y
CONFIG_EXT2_FS_XATTR=y
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT2_FS_SECURITY=y
# CONFIG_EXT2_FS_XIP is not set
CONFIG_EXT3_FS=m
CONFIG_EXT3_FS_XATTR=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT3_FS_SECURITY=y
CONFIG_JBD=m
# CONFIG_JBD_DEBUG is not set
CONFIG_FS_MBCACHE=y
CONFIG_REISERFS_FS=m
# CONFIG_REISERFS_CHECK is not set
CONFIG_REISERFS_PROC_INFO=y
CONFIG_REISERFS_FS_XATTR=y
CONFIG_REISERFS_FS_POSIX_ACL=y
CONFIG_REISERFS_FS_SECURITY=y
CONFIG_JFS_FS=m
CONFIG_JFS_POSIX_ACL=y
CONFIG_JFS_SECURITY=y
# CONFIG_JFS_DEBUG is not set
# CONFIG_JFS_STATISTICS is not set
CONFIG_FS_POSIX_ACL=y
CONFIG_XFS_FS=m
CONFIG_XFS_EXPORT=y
CONFIG_XFS_QUOTA=y
CONFIG_XFS_SECURITY=y
CONFIG_XFS_POSIX_ACL=y
# CONFIG_XFS_RT is not set
CONFIG_OCFS2_FS=m
CONFIG_MINIX_FS=m
CONFIG_ROMFS_FS=m
CONFIG_INOTIFY=y
CONFIG_QUOTA=y
# CONFIG_QFMT_V1 is not set
CONFIG_QFMT_V2=y
CONFIG_QUOTACTL=y
CONFIG_DNOTIFY=y
CONFIG_AUTOFS_FS=m
CONFIG_AUTOFS4_FS=m
CONFIG_FUSE_FS=m

#
# CD-ROM/DVD Filesystems
#
CONFIG_ISO9660_FS=y
CONFIG_JOLIET=y
CONFIG_ZISOFS=y
CONFIG_ZISOFS_FS=y
CONFIG_UDF_FS=m
CONFIG_UDF_NLS=y

#
# DOS/FAT/NT Filesystems
#
CONFIG_FAT_FS=m
```

```
CONFIG_MSDOS_FS=m
CONFIG_VFAT_FS=m
CONFIG_FAT_DEFAULT_CODEPAGE=437
CONFIG_FAT_DEFAULT_IOCHARSET=“ascii”
# CONFIG_NTFS_FS is not set

#
# Pseudo filesystems
#
CONFIG_PROC_FS=y
CONFIG_PROC_KCORE=y
CONFIG_SYSFS=y
CONFIG_TMPFS=y
CONFIG_HUGETLBFS=y
CONFIG_HUGETLB_PAGE=y
CONFIG_RAMFS=y
CONFIG_RELAYFS_FS=m
CONFIG_CONFIGFS_FS=m

#
# Miscellaneous filesystems
#
# CONFIG_ADFS_FS is not set
CONFIG_AFFS_FS=m
CONFIG_HFS_FS=m
CONFIG_HFSPLUS_FS=m
CONFIG_BEFS_FS=m
# CONFIG_BEFS_DEBUG is not set
CONFIG_BFS_FS=m
CONFIG_EFS_FS=m
CONFIG_CRAMFS=m
CONFIG_SQUASHFS=m
# CONFIG_SQUASHFS_EMBEDDED is not set
CONFIG_SQUASHFS_FRAGMENT_CACHE_SIZE=3
# CONFIG_SQUASHFS_VMALLOC is not set
CONFIG_VXFS_FS=m
# CONFIG_HPFS_FS is not set
CONFIG_QNX4FS_FS=m
CONFIG_SYSV_FS=m
CONFIG_UFS_FS=m

#
# Network File Systems
#
CONFIG_NFS_FS=m
CONFIG_NFS_V3=y
CONFIG_NFS_V3_ACL=y
CONFIG_NFS_V4=y
CONFIG_NFS_DIRECTIO=y
CONFIG_NFSD=m
CONFIG_NFSD_V2_ACL=y
CONFIG_NFSD_V3=y
CONFIG_NFSD_V3_ACL=y
CONFIG_NFSD_V4=y
CONFIG_NFSD_TCP=y
CONFIG_LOCKD=m
CONFIG_LOCKD_V4=y
CONFIG_EXPORTFS=m
CONFIG_NFS_ACL_SUPPORT=m
CONFIG_NFS_COMMON=y
CONFIG_SUNRPC=m
CONFIG_SUNRPC_GSS=m
CONFIG_RPCSEC_GSS_KRB5=m
CONFIG_RPCSEC_GSS_SPKM3=m
# CONFIG_SMB_FS is not set
CONFIG_CIFS=m
# CONFIG_CIFS_STATS is not set
CONFIG_CIFS_XATTR=y
CONFIG_CIFS_POSIX=y
# CONFIG_CIFS_EXPERIMENTAL is not set
CONFIG_NCP_FS=m
```

```
CONFIG_NCPFS_PACKET_SIGNING=y
CONFIG_NCPFS_IOCTL_LOCKING=y
CONFIG_NCPFS_STRONG=y
CONFIG_NCPFS_NFS_NS=y
CONFIG_NCPFS_OS2_NS=y
CONFIG_NCPFS_SMALLDOS=y
CONFIG_NCPFS_NLS=y
CONFIG_NCPFS_EXTRAS=y
CONFIG_CODA_FS=m
# CONFIG_CODA_FS_OLD_API is not set
# CONFIG_AFS_FS is not set
CONFIG_9P_FS=m

#
# Partition Types
#
CONFIG_PARTITION_ADVANCED=y
# CONFIG_ACORN_PARTITION is not set
CONFIG_OSF_PARTITION=y
CONFIG_AMIGA_PARTITION=y
# CONFIG_ATARI_PARTITION is not set
CONFIG_MAC_PARTITION=y
CONFIG_MSDOS_PARTITION=y
CONFIG_BSD_DISKLABEL=y
CONFIG_MINIX_SUBPARTITION=y
CONFIG_SOLARIS_X86_PARTITION=y
CONFIG_UNIXWARE_DISKLABEL=y
# CONFIG_LDM_PARTITION is not set
CONFIG_SGI_PARTITION=y
# CONFIG_ULTRIX_PARTITION is not set
CONFIG_SUN_PARTITION=y
CONFIG_KARMA_PARTITION=y
CONFIG_EFI_PARTITION=y

#
# Native Language Support
#
CONFIG_NLS=y
CONFIG_NLS_DEFAULT="utf8"
CONFIG_NLS_CODEPAGE_437=y
# CONFIG_NLS_CODEPAGE_737 is not set
# CONFIG_NLS_CODEPAGE_775 is not set
# CONFIG_NLS_CODEPAGE_850 is not set
# CONFIG_NLS_CODEPAGE_852 is not set
# CONFIG_NLS_CODEPAGE_855 is not set
# CONFIG_NLS_CODEPAGE_857 is not set
# CONFIG_NLS_CODEPAGE_860 is not set
# CONFIG_NLS_CODEPAGE_861 is not set
# CONFIG_NLS_CODEPAGE_862 is not set
# CONFIG_NLS_CODEPAGE_863 is not set
# CONFIG_NLS_CODEPAGE_864 is not set
# CONFIG_NLS_CODEPAGE_865 is not set
# CONFIG_NLS_CODEPAGE_866 is not set
# CONFIG_NLS_CODEPAGE_869 is not set
# CONFIG_NLS_CODEPAGE_936 is not set
# CONFIG_NLS_CODEPAGE_950 is not set
# CONFIG_NLS_CODEPAGE_932 is not set
# CONFIG_NLS_CODEPAGE_949 is not set
# CONFIG_NLS_CODEPAGE_874 is not set
# CONFIG_NLS_ISO8859_8 is not set
# CONFIG_NLS_CODEPAGE_1250 is not set
# CONFIG_NLS_CODEPAGE_1251 is not set
CONFIG_NLS_ASCII=y
CONFIG_NLS_ISO8859_1=m
# CONFIG_NLS_ISO8859_2 is not set
# CONFIG_NLS_ISO8859_3 is not set
# CONFIG_NLS_ISO8859_4 is not set
# CONFIG_NLS_ISO8859_5 is not set
# CONFIG_NLS_ISO8859_6 is not set
# CONFIG_NLS_ISO8859_7 is not set
# CONFIG_NLS_ISO8859_9 is not set
```

```
# CONFIG_NLS_ISO8859_13 is not set
# CONFIG_NLS_ISO8859_14 is not set
# CONFIG_NLS_ISO8859_15 is not set
# CONFIG_NLS_KOI8_R is not set
# CONFIG_NLS_KOI8_U is not set
CONFIG_NLS_UTF8=m

#
# Instrumentation Support
#
# CONFIG_PROFILING is not set
# CONFIG_KPROBES is not set

#
# Kernel hacking
#
# CONFIG_PRINTK_TIME is not set
CONFIG_MAGIC_SYSRQ=y
CONFIG_DEBUG_KERNEL=y
CONFIG_LOG_BUF_SHIFT=17
CONFIG_DETECT_SOFTLOCKUP=y
CONFIG_SCHEDSTATS=y
CONFIG_DEBUG_SLAB=y
# CONFIG_DEBUG_SLAB_LEAK is not set
CONFIG_DEBUG_MUTEXES=y
CONFIG_DEBUG_SPINLOCK=y
CONFIG_DEBUG_SPINLOCK_SLEEP=y
# CONFIG_DEBUG_KOBJECT is not set
CONFIG_DEBUG_BUGVERBOSE=y
CONFIG_DEBUG_INFO=y
CONFIG_DEBUG_FS=y
# CONFIG_DEBUG_VM is not set
# CONFIG_FRAME_POINTER is not set
# CONFIG_FORCED_INLINING is not set
CONFIG_BOOT_DELAY=y
# CONFIG_RCU_TORTURE_TEST is not set
CONFIG_EARLY_PRINTK=y
# CONFIG_DEBUG_STACKOVERFLOW is not set
# CONFIG_DEBUG_STACK_USAGE is not set
# CONFIG_DEBUG_PAGEALLOC is not set
CONFIG_DEBUG_RODATA=y
# CONFIG_4KSTACKS is not set
CONFIG_X86_FIND_SMP_CONFIG=y
CONFIG_X86_MPPARSE=y

#
# Security options
#
CONFIG_KEYS=y
CONFIG_KEYS_DEBUG_PROC_KEYS=y
CONFIG_SECURITY=y
CONFIG_SECURITY_NETWORK=y
CONFIG_SECURITY_NETWORK_XFRM=y
CONFIG_SECURITY_CAPABILITIES=m
CONFIG_SECURITY_SECLVL=m
# CONFIG_SECURITY_SELINUX is not set
CONFIG_SECURITY_TIFPS=m

#
# Cryptographic options
#
CONFIG_CRYPTO=y
CONFIG_CRYPTO_HMAC=y
CONFIG_CRYPTO_NULL=m
CONFIG_CRYPTO_MD4=m
CONFIG_CRYPTO_MD5=y
CONFIG_CRYPTO_SHA1=y
CONFIG_CRYPTO_SHA256=m
CONFIG_CRYPTO_SHA512=m
CONFIG_CRYPTO_WP512=m
CONFIG_CRYPTO_TGR192=m
```

```
CONFIG_CRYPTO_DES=m
CONFIG_CRYPTO_BLOWFISH=m
CONFIG_CRYPTO_TWOFISH=m
CONFIG_CRYPTO_SERPENT=m
CONFIG_CRYPTO_AES=m
# CONFIG_CRYPTO_AES_586 is not set
CONFIG_CRYPTO_CAST5=m
CONFIG_CRYPTO_CAST6=m
CONFIG_CRYPTO_TEA=m
CONFIG_CRYPTO_ARC4=m
CONFIG_CRYPTO_KHAZAD=m
CONFIG_CRYPTO_ANUBIS=m
CONFIG_CRYPTO_DEFLATE=m
CONFIG_CRYPTO_MICHAEL_MIC=m
CONFIG_CRYPTO_CRC32C=m
# CONFIG_CRYPTO_TEST is not set
CONFIG_CRYPTO_SIGNATURE=y
CONFIG_CRYPTO_SIGNATURE_DSA=y
CONFIG_CRYPTO_MPILIB=y

#
# Hardware crypto devices
#
# CONFIG_CRYPTO_DEV_PADLOCK is not set

#
# Library routines
#
CONFIG_CRC_CCITT=m
CONFIG_CRC16=m
CONFIG_CRC32=y
CONFIG_LIBCRC32C=m
CONFIG_ZLIB_INFLATE=y
CONFIG_ZLIB_DEFLATE=m
CONFIG_GENERIC_HARDIRQS=y
CONFIG_GENERIC_IRQ_PROBE=y
CONFIG_GENERIC_PENDING_IRQ=y
CONFIG_X86_SMP=y
CONFIG_X86_HT=y
CONFIG_X86_BIOS_REBOOT=y
CONFIG_X86_TRAMPOLINE=y
CONFIG_X86_SYSENTER=y
CONFIG_KTIME_SCALAR=y
```

B. EMACS .EMACS CONFIGURATION FILE

```
(custom-set-variables
  ;; custom-set-variables was added by Custom -- don't edit or cut/paste it!
  ;; Your init file should contain only one such instance.
 '(auto-compression-mode t nil (jka-compr))
 '(case-fold-search t)
 '(current-language-environment “UTF-8”)
 '(default-input-method “rfc1345”)
 '(global-font-lock-mode t nil (font-lock))
 '(show-paren-mode t nil (paren)))
(custom-set-faces
  ;; custom-set-faces was added by Custom -- don't edit or cut/paste it!
  ;; Your init file should contain only one such instance.
 )

(defun linux-c-mode ()
  “C mode with adjusted defaults for use with the Linux kernel.”
  (interactive)
  (c-mode)
  (c-set-style “K&R”)
  (setq tab-width 8)
  (setq indent-tabs-mode t)
  (setq c-basic-offset 8))

(setq auto-mode-alist (cons '(“.*\\.[ch]$” . linux-c-mode)
                            auto-mode-alist))
```

LIST OF REFERENCES

1. Afinidad, F. B., Levin, T. E., Irvine, C. E., Nguyen, T. D., “A Model for Temporal Interval Authorizations,” Hawaii International Conference on System Sciences, Software Technology Track, Information Security Education and Foundational Research, Kauai, Hawaii, January 2006.

2. Afinidad, F. B., Levin, T. E., Irvine, C. E., Nguyen, T. D., “A Time Interval Memory Protection System,” Secure Core Technical Report, 2006.

3. Afinidad, F. B., An Interval Algebra-Based Temporal Access Control Protection Architecture, Ph.D. Dissertation, Naval Postgraduate School, Monterey, California, June 2005.

4. Rule Set Based Access Control, http://www.rsbac.org/documentation/why_rsbac_does_not_use_lsm, July 2006.

5. Grsecurity, http://grsecurity.net/lsm.php, July 2006.

6. Subversion, http://subversion.tigris.org, September 2006.

7. Source Insight Program Editor and Analyzer, http://www.sourceinsight.com, September 2006.

8. Fedora Core 5 Linux Distribution, http://fedora.redhat.com, September 2006.

9. Weissman, C., Security Controls in the ADEPT-50 Time-Sharing System. Proceedings of the Fall Joint Computer Conference, November 18-20 (1969) 119-133.

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

Gorman, M. “Understanding the Linux Virtual Memory Manager,” *Bruce Peren’s Open Source Series,* Prentice Hall, 2004.

Bovet, D. P., Cesati, M., *Understanding the Linux Kernel*, 2nd ed., O’Reilly, Sebastopol, 2003.

Love, R., *Linux Kernel Development*, 2nd ed., Novell Press, Indianapolis, 2005.

Morris, J., “Filesystem Labeling in SELinux,” Linux Journal, November 2004, http://www.linuxjournal.com/article/7689, September 2006.

Source Code for Linux, http://lxr.linux.no/source/, August 2006.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Hugo A. Badillo
   NSA
   Fort Meade, Maryland

4. George Bieber
   OSD
   Washington, DC

5. JohnCampbell
   NSA
   Fort Meade, Maryalnd

6. Deborah Cooper
   DC Associates, LLC
   Roslyn, Virginia

7. CDR Daniel L. Currie
   PMW 161
   San Diego, California

8. Louise Davidson
   National Geospatial Agency
   Bethesda, Maryland

9. Steve Davis
   NRO
   Chantilly, Virginia

10. Vincent J. DiMaria
    National Security Agency
    Fort Meade, Maryland

11. CDR James Downey
    NAVSEA
    Washington, DC

12. Dr. Diana Gant
National Science Foundation
Arlington, Virginia

13. Jennifer Guild
SPAWAR
Charleston, South Carolina

14. Richard Hale
DISA
Falls Church, Virginia

15. CDR Scott D. Heller
SPAWAR
San Diego, California

16. Wiley Jones
OSD
Washington, DC

17. Russell Jones
N641
Arlington, Virginia

18. David Ladd
Microsoft Corporation
Redmond, Washington

19. Dr. Carl Landwehr
DTO
Fort George T. Meade, Maryland

20. Steve LaFountain
NSA
Fort Meade, Maryland

21. Dr. Greg Larson
IDA
Alexandria, Virginia

22. Dr. Karl Levitt
NSF
Arlington, Virginia

23. Dr. Vic Maconachy
NSA
Fort Meade, Maryland

24. Doug Maughan
Department of Homeland Security
Washington, DC

25. Dr. John Monastra
Aerospace Corporation
Chantilly, Virginia

26. John Mildner
SPAWAR
Charleston, South Carolina

27. Mark T. Powell
Federal Aviation Administration
Washington, DC

28. Jim Roberts
Central Intelligence Agency
Reston, Virginia

29. Jon Rolf
NSA
Fort Meade, Maryland

30. Ed Schneider
IDA
Alexandria, Virginia

31. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

32. Charles Sherupski
Sherassoc
Round Hill, Virginia

33. Ken Shotting
NSA
Fort Meade, Maryland

34. CDR Wayne Slocum
SPAWAR
San Diego, California

35. Dr. Ralph Wachter
ONR
Arlington, Virginia

36. David Wirth
N641
Arlington, Virginia

37. CAPT Robert Zellmann
CNO Staff N614
Arlington, Virginia

38. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, California

39. Thuy D. Nguyen
Naval Postgraduate School
Monterey, California

40. Ken Chiang
Affiliation (SFS students: Civilian, Naval Postgraduate School)
Monterey, California